jamk.fi

Palvelun laatu SDN-verkossa

Samu Nerg

Opinnäytetyö
Toukokuu 2016
Tekniikan ja liikenteen ala
Insinööri (AMK), tietotekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Nerg, Samu	Opinnäytetyö, AMK	21.5.2016
	Sivumäärä 105	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Palvelun laatu SDN-verkossa

Tutkinto-ohjelma
Tietotekniikan koulutusohjelma

Työn ohjaaja(t)
Jarmo Viinikanoja, Mika Rantonen

Toimeksiantaja(t)
Jyväskylän Ammattikorkeakoulu, Cyber Trust –projekti, Karo Saharinen

Tiivistelmä

Cyber Trust on Digilen tutkimusohjelma, jonka tavoitteena on nostaa Suomi johtavaksi kyberturvallisuuden maaksi maailmassa lisäämällä yritysten, yliopistojen ja tutkimuslaitosten välistä yhteistyötä.

Opinnäytetyön toimeksiantona oli tutkia mahdollisuuksia parantaa palvelun laatua SDN-verkoissa hyödyntäen avoimen lähdekoodiin kontrollereita ja kytkimiä. Opinnäytetyön oli myös tarkoitus laajentaa Cyber Trust -projektin SDN-testbed ympäristöä fyysisille verkkolaitteille. Testiympäristön laajennuksen oli tarkoitus mahdollistaa SDN-aiheisten laborato-rioharjoitusten toteutus.

Opinnäytetyön toteutuksessa hyödynnettiin kolmea Raspberry Piitä, joita käytettiin SDN-verkon kytkiminä hyödyntäen Open vSwitch-virtuaalikytkimiä. Raspberry Piiden hallinta toteutettiin käyttämällä RYU-kontrolleria. Raspberry Piiden portteihin konfiguroitiin erilaisia jonoja eri liikennevirroille. Liikennevirrat merkattiin ja ohjattiin eteenpäin käyttäen RYU-kontrolleria. Liikennevirtoja lähetettiin JDSUn liikennegeneraattorilla mittalaitteisiin, joiden avulla saatiin tulostettua raportteja, joita analysoimalla saatiin tietoa verkon käyttäytymisestä ruuhkatilanteissa erilaisilla palvelunlaatuasetuksilla. Raspberry Piit liitettiin lopuksi osaksi JYVSECTECin RGCE-ympäristöä. Tätä laajennusta on tarkoitus jatkossa hyödyntää SDN-tekniikoiden tutkimus– ja koulutuskäytössä.

SDN-verkkojen palvelun laadun tutkimus onnistui hyvin. Toteutuksen aikana löydettiin toimivia tapoja parantaa SDN-verkon palvelun laatua kuin myös tärkeää tietoa puutteista, joita avoimen lähdekoodin ohjelmistot sisältävät. SDN-testbed-ympäristön laajennus toteutettiin, mutta tiettyjen ohjelmistojen kanssa ilmenneitä ongelmia ei ratkaistu.

Avainsanat (asiasanat)
SDN, OpenFlow, Palvelun laatu, Raspberry Pi, RYU, ONOS, OVS, RGCE

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Nerg, Samu	Bachelor's thesis	21.5.2016
		Language of publication: finnish
	Number of pages 105	Permission for web publication: x

Title of publication
Quality of Service in Software Defined Networking

Degree programme
Information Technology

Supervisor(s)
Jarmo Viinikanoja, Mika Rantonen

Assigned by
JAMK University of Applied Sciences, Cyber trust -project, Karo Saharinen

Abstract

The thesis was assigned by Cyber Trust project in JAMK University of Applied Sciences. Cyber Trust is a Digile research program with the goal of making Finland a leading country in the field of cyber security by adding cooperation between enterprises, universities and research facilities.

The assignment was to research ways to improve quality of service in SDN by using open source controllers and switches. The Thesis was also set out to expand Cyber Trust project's SDN testbed environment to physical devices. The expanded test environment was to enable execution of SDN laboratory exercises.

The thesis was executed with three Raspberry Pis used as of-switches using Open vSwitch virtual switches. The control of Raspberry Pis was implemented with RYU-controller. Different types of queues were configured to Raspberry Pi's physical ports for different datastreams. The data streams were marked and forwarded using RYU-controller and created with JDSU traffic generator to measuring devices, which enabled to create reports. Data about the behavior of the network in congested state was gathered by analyzing those reports. At the end Raspberry Pis were implemented as a part of JYVSECTECs RGCE environment. This implementation is to be used for SDN research and education.

The research on quality of service in SDN was successful. During the research, good ways to improve quality of service in SDN were found as well as information about flaws in open source products. The expanding of SDN testbed was done, however, occurring problems with certain software could not be solved.

Keywords/tags (subjects)
SDN, OpenFlow, Quality of Service, Raspberry Pi, RYU, ONOS, OVS, RGCE

Miscellaneous

Sisältö

Kuviot

Taulukot

Lyhenteet

ACL	Access Control List
AF	Assured Forwarding
API	Application Programming Interface
AS	Autonomous System
BA	Behavior Aggregate
BE	Best Effort
BGP	Border Gateway Protocol
CBQ	Class Based Queuing
CPSI	Control Plane Southbound Interface
CS	Class Selector
DiffServ	Differentiated Services
DNS	Domain Name System
DSCP	Differentiated Services Code Point
ECN	Explicit Congestion Notification
EF	Expedited Forwarding
FIFO	First In First Out
HFSC	Hierarchial Fair-Service Curve
HTB	Hierarchial Token Bucket
ICMP	Internet Control Message Protocol
IntServ	Integrated Services
IP	Internet Protocol
JYVSECTEC	Jyväskylä Security Technology
MBZ	Must Be Zero
MAC	Media Access Control
MPLS	Multiprotocol Label Switching
MPSI	Management Plane Southbound Interface
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
NETCONF	Network Configuration Protocol
NFV	Network Function Virtualization
OF	OpenFlow
ONF	Open Networking Foundation
ONOS	Open Network Operating System
OSPF	Open Shortest Path First
OVS	Open Virtual Switch
OVSDB	Open Virtual Switch Database
PIP	Pip Installs Python
PHB	Per-Hop Behavior
PKI	Public Key Infrastructure
QoS	Quality of Service
RSVP	Resource Reservation Protocol

RGCE	Realistic Global Cyber Environment
TCP	Transmission Control Protocol
TLS	Transport Layer Security
ToS	Type of Service
SLA	Service Level Agreement
SDN	Software Defined Networking
STP	Spanning Tree Protocol
VoIP	Voice over Internet Protocol
WAN	Wide Area Network
YANG	Yet Another Next Generation

1 Työn lähtökohdat

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi Jyväskylän ammattikorkeakoulu. JAMK on vetovoimainen ja kansainvälinen korkeakoulu, jossa on opiskelijoita kokonaisuudessaan yli 8500 sekä henkilökuntaa noin 700 henkilön verran. JAMKin yksiköitä ovat ammatillinen opettajakorkeakoulu, hyvinvointiyksikkö, liiketoimintayksikkö sekä teknologiayksikkö. JAMKilla on useita toimipisteitä Jyväskylässä sekä yksi toimipiste Saarijärven Tarvaalassa. JAMKissa on mahdollista suorittaa yli 30 erilaista tutkintoa kahdeksalta eri alalta. Kansainvälisyys JAMKissa näkyy mm. vaihto-opiskelijoiden määrässä, joita on yli 70 maasta (Tietoa JAMKista n.d.)

Jyväskylän ammattikorkeakoulusta työn tilaajana toimi Cyber Trust –projekti. Cyber Trust on Digilen rahoittama hanke, joka on käynnistynyt toukokuussa 2015. Cyber Trustin tavoitteena on olla nostamassa Suomea kyberturvallisuuden edelläkävijäksi maailmassa ja lisätä yritysten, yliopistojen ja tutkimuslaitosten välistä yhteistyötä. Tällä tavoin saavutetaan parempi kansainvälinen kilpailukyky ja uusia liiketoimintamahdollisuuksia. Cyber Trust –projektissa on mukana 21 yksityistä yritystä ja 9 tutkimuslaitosta. Ohjelman suunniteltu kesto on kolme ja puoli vuotta (Vainionkulma-Immonen, 2015.)

Cyber Trust –projekti on jaettu neljään työryhmään. JAMK osallistuu näistä työryhmään kolme. Sen tavoitteena on uusien verkkoteknologien tietoturvallisuuden tutkiminen ja kehittäminen. JAMKin tutkimustapauksen omistajana projektissa toimii Elisa. Tästä syystä JAMK keskittyy projektissa tutkimaan uusien verkkoteknologioiden tuomia mahdollisuuksia ja uhkia verkko-operaattorin näkökulmasta (Savola 2014.)

1.2 Toimeksianto ja työn tavoitteet

Opinnäytetyön toimeksiantona oli tutkia, miten palvelun laatua pystytään toteuttamaan SDN-verkossa avoimen lähdekoodin SDN-ohjelmistoja hyödyntäen. Työn teoriaosuudessa käsitellään SDN-arkkitehtuuria, OF-protokollaa sekä NFV-arkkitehtuuria verrattuna perinteisiin tietoverkkoihin. Palvelun laatua käsitellään tarkastellen palvelun laadun parametreja, mittausmenetelmiä ja siihen liittyviä toteutusmekanismeja tarkastellaan yleisellä tasolla.

Työn toteutuksessa oli tavoitteena löytää palvelun laadun toteutukseen sopivia ohjelmia ja testata niiden käytännön toimivuutta. Palvelun laadun toteutumista haluttiin todentaa erilaisilla mittauksilla, jossa verkkoon generoidaan ruuhkatilanne, ja verkon käyttäytymistä analysoidaan mittalaitteilla. Työn toimeksiantaja halusi myös laajennusta heillä ennestään käytössä olevaan virtualisoituun SDN-testbed ympäristöön. Ympäristöön tehtiin JAMKin omana investointina laajennus fyysisillä laitteilla. Laajennuksen tarkoitus oli mahdollistaa SpiderNet-ympäristön liittäminen SDN-testbed-ympäristöön. Molempien ympäristöjen liitettävyys keskenään antaa uusia mahdollisuuksia tutkimukselle.

2 Software Defined Networking

2.1 Yleistä

SDN tarjoaa uuden lähestymistavan tietoverkon ohjelmointiin. Se tarkoittaa kykyä alustaa, kontrolloida, muuttaa ja hallinnoida verkon käyttäytymistä dynaamisesti. SDN:ssä erilaiset applikaatiot pystyvät ohjelmoimaan erillisiä verkkolaitteita ja näin ollen kontrolloimaan verkkoa kokonaisuutena keskitetysti yhteen pisteeseen. SDN koostuu useista tekniikoista, joiden avulla verkkopalveluita voidaan toteuttaa mukautuvalla, dynaamisella ja skaalautuvalla tavalla (Denazis, Hadi Salim, Haleplidis, Koufopavlou, Meyer & Pentikoudis 2015.)

ONF on voittoa tavoittelematon käyttäjäkeskeinen organisaatio, jonka tavoitteena on kiihdyttää SDN-verkkojen käyttöönottoa. ONF:ssä on yli 100 jäsenyritystä pienistä startup-yrityksistä kansainvälisiin markkinajohtajiin. Tunnetumpia ONF:n jäsenyrityksiä ovat mm. Google, Facebook, MicrosOFt, Yahoo ja Deutsche Telekom. ONF keskittyy kehittämään SDN-tekniikoita avointen standardien kanssa. ONF:n merkittävin standardi on OF, josta on tullut yleisin käytössä oleva protokolla SDN-kontrollerin ja verkkolaitteiden väliseen kommunikaatioon (OF n.d.b.)

2.2 Syyt muutokselle

Perinteinen tietoverkkoarkkitehtuuri on monella tapaa vanhanaikainen täyttämään nykypäivän vaatimuksia yritysverkkojen, operaattoriverkkojen ja asiakasverkkojen osalta. Internetissä olevien laitteiden ja liikenteen määrä on kasvanut räjähdysmäisesti etenkin lisääntyneiden mobiililaitteiden myötä. Myös palvelinten virtualisointi ja pilvipalveluiden yleistyminen ovat asettaneet uudenlaisia vaatimuksia tietoverkoille. SDN-arkkitehtuurin on määrä tarjota dynaaminen, helposti hallittava, kustannustehokas ja skaalautuva vaihtoehto perinteisille tietoverkoille (Software-Defined Networking: The New Norm for Networks 2012, 2-3.)

Tietoliikennevirrat esimerkiksi konesaliyritysten verkoissa ovat muuttuneet merkittävästi. Verrattuna applikaatioihin, joissa yksi asiakas kommunikoi yhden palvelimen kanssa, nykyään yksi applikaatio voi kommunikoida useiden palvelinten ja tietokantojen kanssa muodostaen monimutkaisempia liikennevirtoja. (Software-Defined Networking: The New Norm for Networks 2012, 3-4.)

Yritysten tarjoamien julkisten ja yrityksen sisäisten pilvipalveluiden määrän kasvu on johtanut siihen, että yrityksen ohjelmistoihin, infrastruktuuriin ja muihin resursseihin on oltava saatavissa yhteys kaiken aikaa. Usein käyttäjillä on tarve päästä yrityksen verkkoon mistä tahansa ja millä tahansa laitteella. Tämän lisäksi pilvipalveluiden tulee olla tietoturvallisia ja dynaamisesti skaalautuvia tietojenkäsittelyn, varastotilan ja verkkoresurssien suhteen nopeasti muuttuvassa ympäristössä. Suurten datakeskus-

ten ja operaattoreiden vaatimukset ovat muuttuneet vuosien mittaan. Verkkolaitteiden, palvelinten, käyttäjien ja tietoliikenteen määrät ovat kasvaneet niin suuriksi, että vanhanaikaiset, manuaalisesti konfiguroitavat verkot eivät skaalaudu tällaiseen mittakaavaan (Software-Defined Networking: The New Norm for Networks 2012, 3-4, 6.)

Perinteiset verkkoarkkitehtuurit ovat perustuneet pitkälti toisistaan irrallisiin protokolliin. Protokollat on kehitetty eristettyinä toisistaan jokaisen protokollan keskittyessä yhden ongelman ratkaisemiseen. Tämän seurauksena tietoverkoista on tullut hyvin monimutkaisia kokonaisuuksia. Verkkoon tehtävät muutokset ovat hitaita ja kömpelöitä, koska yhden laitteen vaihtaminen tai poistaminen verkosta voi vaatia konfiguraatiomuutoksia useissa verkon laiteissa. Tämän lisäksi on otettava huomioon laitevalmistaja, laitteen malli ja ohjelmistoversio, jotta voidaan varmistaa, että laite on yhteensopiva nykyisen verkkotopologian ja sen vaatimien protokollien suhteen. Palvelinten virtualisointi on lisännyt verkkoyhteyden tarvitsemien koneiden määrää merkittävästi ja muuttanut käsitystä koneiden fyysisestä sijainnista verkossa. Applikaatiot voivat pyöriä jaetusti usealla virtuaalikoneella, ja niiden liikennevirrat vaihtelevat dynaamisesti palvelinten tehdessä kuormanjakoa (Software-Defined Networking: The New Norm for Networks 2012, 4-5.)

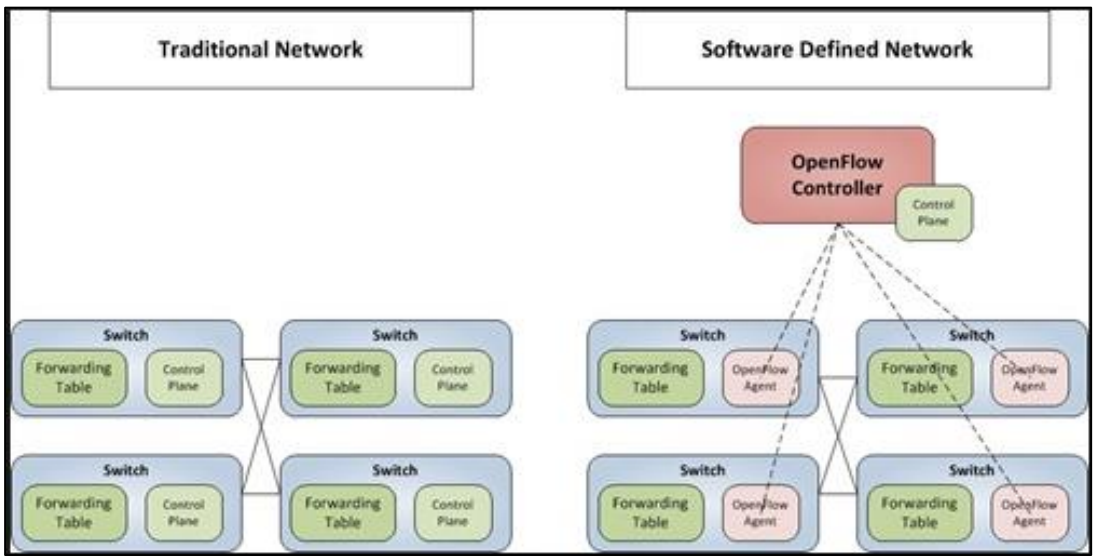
Tietoverkkojen koon ja monimutkaisuuden kasvaessa on yhä vaikeampaa konfiguroida yhdenmukaisia sääntöjä kaikille verkon laitteille. Esimerkiksi yhdenmukaisten ACL-sääntöjen konfigurointi koko verkkoon vie paljon työtunteja, ja yhden koneen lisäys verkkoon saattaa vaatia muutoksia useiden laitteiden konfigurointiin. Yksikin väärin konfiguroitu laite voi aiheuttaa tietoturvariskin yrityksen verkolle. Usein yritysverkoissa voi kulkea tavallisen dataliikenteen lisäksi reaaliaikaista liikennettä äänen ja videon muodossa. Tämän lisäksi verkossa liikkuu erilaisia hallintaprotokollia ja esimerkiksi erilaisten tietokantojen välistä kriittistä dataa. Monimutkaisten tietoliikennevoiden QoS-asetusten konfigurointi verkon päästä päähän on aikaa vievää työtä, ja sitä vaikeuttavat entisestään eroavaisuudet eri laitevalmistajien välillä (Software-Defined Networking: The New Norm for Networks 2012, 5.)

Yritykset ovat tällä hetkellä hyvin riippuvaisia verkkolaitevalmistajista. Yritysten nopeutta reagoida uusiin vaatimuksiin rajoittaa laitevalmistajien hitaus. Laitteiden uusien versioiden tai uusien protokollien saaminen markkinoille voi viedä useita vuosia,

mikä voi olla liikaa nopeasti kehittyvissä ympäristöissä. Samanaikaisesti laitevalmistajien lisenssimaksut ovat korkeita, mikä tuo huomattavia kustannuksia yrityksille (Software-Defined Networking: The New Norm for Networks 2012, 6.)

2.3 Arkkitehtuuri

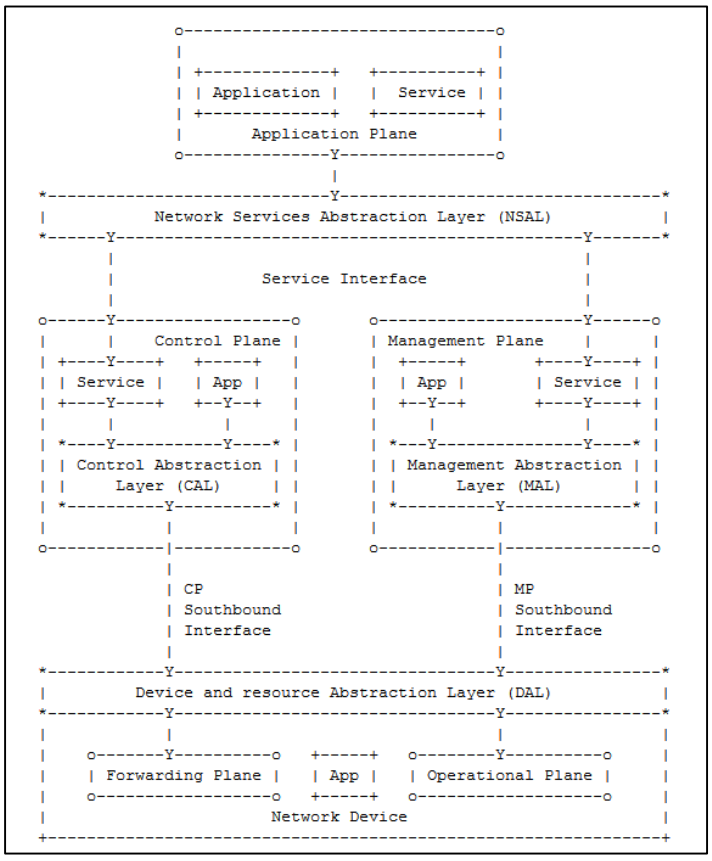
SDN ja perinteinen tietoverkko eroavat ratkaisevasti arkkitehtuuriltaan. Perinteisestä tietoverkosta poiketen SDNssä verkkolaitteiden kontrolli – ja välitystaso pyritään pitämään erillisinä ja kontrollitaso keskitetään yhteen paikkaan. Vastaavasti perinteisissä verkkolaitteissa välitys – ja kontrollitaso ovat samalla tasolla, ja ne käyttävät yhteisiä rajapintoja hyötydatan ja hallintaprotokollien kuljettamiseen. Tasojen erottamisella pyritään vähentämään verkon monimutkaisuutta sekä nopeuttamaan molempien tasojen kehitystyötä. SDN perustuu ajatukseen, jossa verkko koostuu kontrolloitavista laitteista ja kontrolloivista laitteista eli SDN-kontrollereista. Kontrolloiva laite hallitsee kontrolloitavaa laitetta jonkin rajapinnan kautta. Yleensä tämä rajapinta on API. Tämän rajapinnan kautta SDN-kontrollerit hallitsevat verkkolaitetta kontrollitasolla ja kertovat mitä kontrolloitavan laitteen tulee tehdä välitystasolla (Denazis ym. 2015, 6.) Kuviossa 1 esitetään SDN-verkon arkkitehtuuri verrattuna perinteisen tietoverkon arkkitehtuuriin.



Kuvio 1. SDN-verkon vertailu perinteiseen tietoverkkoon (Ghosh 2014)

Kuviossa esitetään, kuinka perinteisessä tietoverkossa jokaisella verkkolaitteella on oma reititys – tai kytkentätaulu. Tämän lisäksi hallintataso on jakautunut jokaiselle verkkolaitteelle erikseen. SDN-verkossa hallintataso on keskitettynä yhdelle tai useammalle SDN-kontrollerille, jotka kommunikoivat kytkimen OF-agentin kanssa. Tällä tavoin verkon älykkyys on siirretty pois verkkolaitteilta, joiden ainoaksi tehtäväksi jää pakettien uudelleenohjaus välitystasolla (Ghosh 2014.)

Kokonaisuudessaan SDN-arkkitehtuuri koostuu seuraavista tasoista: välitystaso (Data Plane), toimintataso (Operational Plane), kontrollitaso (Control Plane), hallintataso (Management Plane) ja sovellustaso (Application Plane) (Denazis ym. 2015, 8-9.) Arkkitehtuuri eri tasoineen esitetään kuviossa 2.



Kuvio 2. SDN-arkkitehtuuri (Denazis ym. 2015, 7)

Kuvassa alimmalla tasolla ovat välitys – ja toimintataso, jotka sijaitsevat verkkolaitteella. Verkkolaitteella on rajapinta SDN-kontrolleriin, joka hallitsee verkkolaitetta kontrolli – ja hallintatasolla. Näillä tasoilla on puolestaan rajapinta sovellustasolle, josta verkon kokonaisuutta hallinnoidaan. Eri tasojen välillä on erotuskerrokset (Abstraction Layer), ja eri tasojen ja kerrosten välisiä rajapintoja on merkitty merkillä Y (Denazis ym. 2015, 8-9.)

Välitystasolla käsitellään tietopolulla kulkevia paketteja kontrollitasolta saatujen ohjeiden perusteella. Välitystason toimintoja ovat pakettien uudelleenohjaus, pudottaminen ja muokkaus. Välitystaso on yleensä päätepiste kontrollitason palveluille ja applikaatioille. Toimintataso hallinnoi verkkolaitteen toiminnallista tilaa, kuten käytössä olevia portteja ja niiden tilaa, muistia ja prosessoritehoa. Toimintataso on päätepiste hallintatason palveluille ja applikaatioille. Toimintatasoa ei aina mielletä erilliseksi tasoksi, vaan sitä pidetään osana välitystason toiminnollisuutta (Denazis ym. 2015, 8-9.)

Kontrollitaso tekee päätökset siitä, mitä verkkolaitteille saapuville paketeille kuuluu tehdä. Kontrollitaso keskittyy enimmäkseen päätöksentekoon välitystasolla eikä niinkään toimintatasolla. Kontrollitaso saattaa kuitenkin tarvita toimintatasolta tietoja esimerkiksi käytössä olevista porteista. Kontrollitason tärkein tehtävä on muokata verkkolaitteiden vuotauluja verkkotopologian ja palvelupyyntöjen perusteella. Hallintatasolla monitoroidaan, konfiguroidaan ja ylläpidetään verkkolaitteita esimerkiksi tekemällä muutoksia verkkolaitteen tilaan. Hallintataso keskittyy enemmän verkkolaitteen toimintatasolle kuin välitystasolle. Sovellustasolla toimivat sovellukset ja palvelut määrittelevät tietoliikenteen käyttäytymisen verkossa. Sovelluksia, jotka vaikuttavat suoraan välitystason toimintaan kuten reitityspäätöksiin, ei pidetä osana sovellustasoa vaan kontrollitasoa. Sovellukset voivat olla modulaarisia ja jakautettuja ja voivat siten olla yhteydessä useaan tasoon (Mts. 9-10.)

2.4 SDN-kontrolleri

SDN-arkkitehtuurin älykkyys on keskitetty palvelimelle, jota kutsutaan kontrolleriksi. Kontrolleri hallitsee verkkolaitteiden vuotauluja eteläsuuntaisen (Southbound) API:n kautta ja sovelluksia pohjoissuuntaisen (Northbound) API:n kautta. Kontrolleri sisältää useimmiten useita erillisiä moduuleita, joista jokainen toteuttaa erillistä verkon toiminnallisuutta. Esimerkiksi yksi moduuli voi toimia reitittävänä komponenttina ja toinen voi toteuttaa palomuurausta. Kontrollerin tehtäviin kuuluu pitää yllä tietoa verkkolaitteista, käytössä olevissa porteista ja päätelaitteista. Näiden tietojen perusteella kontrolleri saa verkosta kokonaiskuvan, jonka perusteella se pystyy dynaamisesti asettamaan uusia voita verkkolaitteiden vuotauluihin. Kontrollereiden eniten käyttämiä protokollia ovat OF, OVSDB, YANG ja NetConf. Useille kontrollereille on tehty myös sovelluksia, joiden avulla ne voivat konfiguroida vanhoja verkkoprotokollia kuten OSPF, MPLS ja BGP (What are SDN Controllers (or SDN Controllers Platforms)? n.d.)

3 OpenFlow

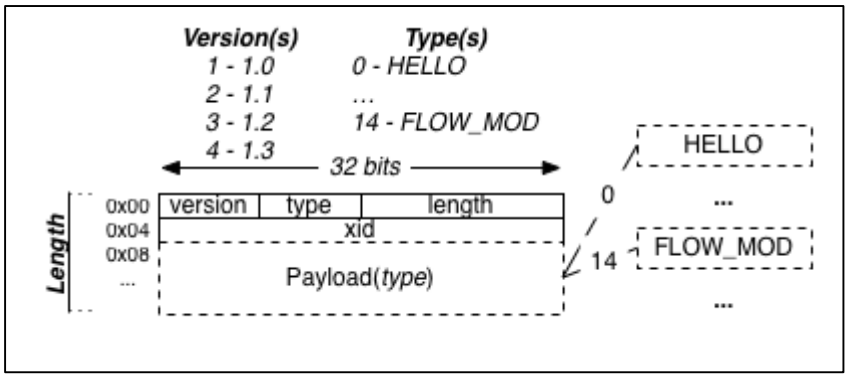
3.1 Yleistä

OpenFlow on ONF:n kehittämä ensimmäinen standardoitu rajapinta SDN-arkkitehtuurin hallinta – ja välitystasojen väliseen kommunikointiin. OpenFlow mahdollistaa SDN-arkkitehtuurin vaatimukset eristetyistä hallinta – ja välitystasoista, keskitetystä verkon hallinnasta ja ohjelmoitavuudesta. OpenFlowin avulla OpenFlow-kytkinten vuotauluja pystytään muokkaamaan dynaamisella tavalla (Openflow n.d.a.)

3.2 Arkkitehtuuri

3.2.1 OpenFlow-otsikko

OpenFlow-protokolla voidaan jakaa neljään komponenttiin: viestikerros (message layer), tilakone (state machine), järjestelmärajapinta (system interface) ja konfiguraatio (configuration). Viestikerros on OF-protokollapinon ydin. Se määrittelee OF-viestien rakenteen. Viestikerros mahdollistaa viestien muodostamisen, kopioinnin, vertailun, tulostamisen ja muokkauksen. OF-viestin otsikkokenttä esitetään kuviossa 3 (Openflow n.d.a.)

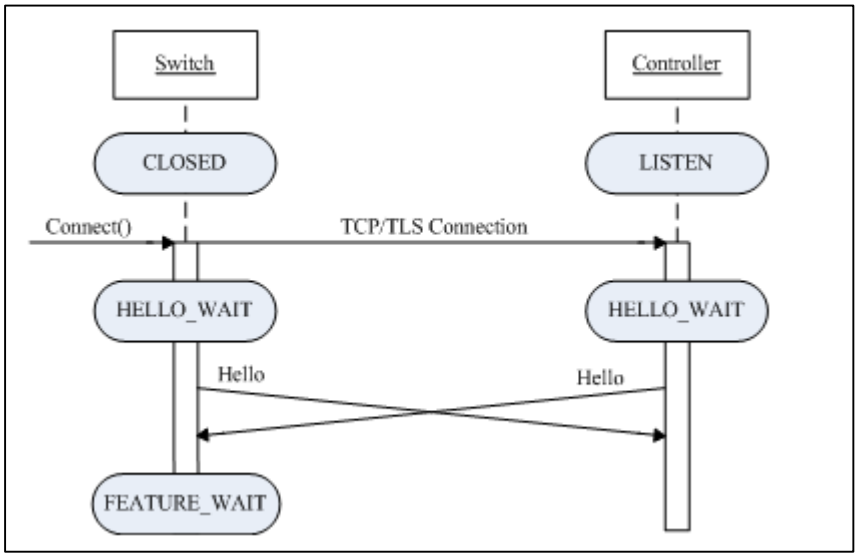


Kuvio 3. OF-otsikko (OF n.d.a)

Jokainen OF-viesti alkaa samalla otsikkorakenteella, ja se on riippumaton OF-versiosta. Otsikko alkaa version-kentällä, joka kertoo, mitä OF-versiota paketti on. Type-kenttä kuvaa, mitä toimintoa OF-paketti suorittaa. Se voi olla esimerkiksi hello-viesti kontrolliyhteyden ylläpitämiseksi, vuon lisääminen vuotauluun tai konfiguraatiotietojen hakuviesti. Lenght-kenttä kertoo, kuinka monta bittiä paketin pituus on. Xid-kenttä sisältää uniikin tunnisteen, jonka perusteella pyynnöt ja niiden vastaukset voidaan tunnistaa. Näiden otsikkokenttien jälkeen alkaa hyötydata, joka kulkee TCP-paketin sisällä (Openflow n.d.a.)

3.2.2 OpenFlow-yhteyden muodostus

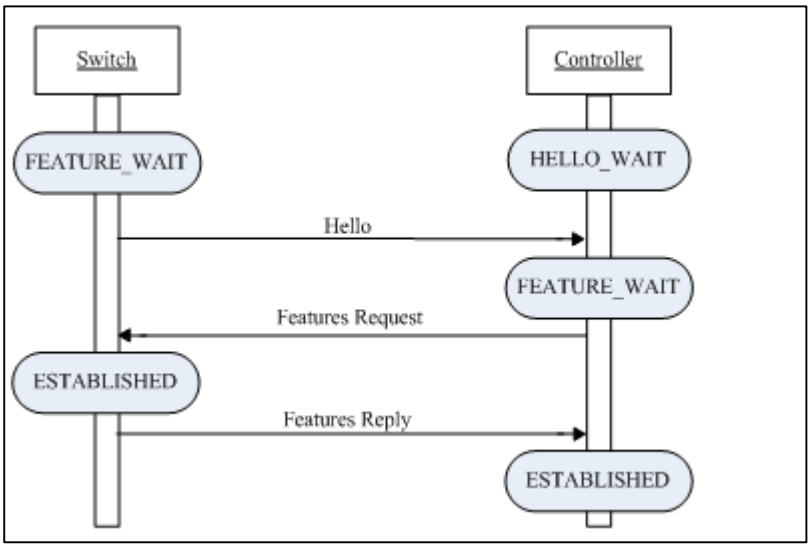
OpenFlow on yksinkertaistettu protokolla, jonka lähes kaikki viestit ovat asynkronisia, eli ne eivät vaadi tilatietoja. Tilakonetta tarvitaan ainoastaan OF-kytkimen ja OF-kontrollerin välisen yhteyden muodostamiseen. Yhteyden muodostaminen koostuu kahdesta vaiheesta: OF-version neuvottelusta ja ominaisuuksien tiedostelusta. Heti alemman tason TCP/TLS-yhteyden muodostumisen jälkeen kytkin ja kontrolleri neuvottelevat, mitä OF-versiota laitteet käyttävät kommunikointiin. Mikäli laitteet eivät löydä OF-versiota, jota molemmat tukevat, ei yhteyden ylläpitäminen ole mahdollista (Openflow n.d.a.) Kuviossa 4 esitetään onnistunut yhteyden muodostus OF-kytkimen ja OF-kontrollerin välillä.



Kuvio 4. OF-yhteyden onnistunut muodostus (Openflow n.d.a)

Lähtötilanteessa kontrolleri kuuntelee OF-porttia, joka on useimpien ohjelmistojen vakioasetuksilla 6633. Kytkin yrittää muodostaa TCP/TLS-yhteyden kontrolleriin. Yhteyden muodostamisen jälkeen kytkin ja kontrolleri lähettävät toisilleen hello-viestit, joiden perusteella laitteet tietävät, mitä OF-versiota vastakkainen laite yrittää käyttää. OF-versioissa ennen 1.3.1 käytetty versio on vanhin molempien laitteiden tukema versio. Uudemmissa versioissa käytetään uusinta mahdollista OF-versiota. Kun

yhteinen versio on löytynyt, siirrytään ominaisuuksien tiedustelu vaiheeseen (Openflow n.d.a.) Tiedusteluvaihe esitetään kuviossa 5.



Kuvio 5. OF-yhteyden ominaisuuksien tiedustelu vaihe (Openflow n.d.a)

Kontrolleri odottaa ensin kytkimeltä saapuvaa hello-viestiä, jonka jälkeen se lähettää tiedustelun kytkimen ominaisuuksista. Kytkin lähettää vastauksen kontrollerille ja mikäli vastaus saapuu perille, on OF-yhteys muodostettu onnistuneesti (Openflow n.d.a.)

3.3 OpenFlow-kytkin

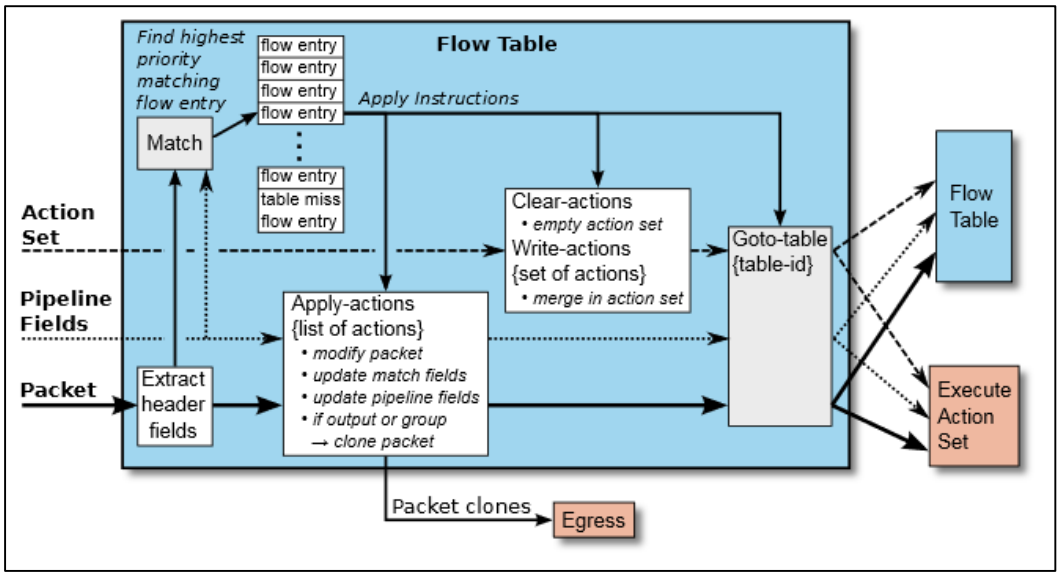
OpenFlow-kytkin muodostuu yhdestä tai useammasta vuotaulusta, ryhmätaulusta ja vähintään yhdestä kanavasta OF-kontrolleriin. Kontrolleri lisää voita kytkimen vuotauluun joko reaktiivisesti saapuvien pakettien mukaan tai proaktiivisesti, jolloin vuot ovat pysyviä. (Appenzeller, Balland, Barker, Beckmann, Casado, Crabbe, Cohn, Curtis, Das, Ding, Dheureuse, Dunbar, Gandham, Erickson, Gibb, Heller, Kobayashi, Lajos Kis, Lantz, Madabushi, Malek, McDysan, McKeown, Mizrahi, Moses, Nygren, Orr, Pettit, Pfaff, Poutievski, Price, Ramanathan, Schneider, Sherwood, Talayco, Takahashi,

Tonsing, Tourrilhes, Vicisano, Ward, Yabe, Yiakoumis, Yadav & Yap. 2014, 11-12.) OF-vuotaulun kentät esitetään taulukossa 1.

Taulukko 1. OF-vuotaulun kentät (Appenzeller ym. 2014, 22)

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Vuotaulu koostuu seitsemästä kentästä. Match field-kenttä määrittelee parametrit, joihin sisään tulevia paketteja verrataan. Parametrina voidaan käyttää esim. fyysistä porttia, MAC – tai IP-osoitetta tai muita otsikkotietoja. Priority-kentällä määritellään voiden tärkeys. Counters-kenttä kertoo, kuinka monta pakettia on käsitelty tietyn vuon mukaan. Instructions-kenttä kertoo mitä paketilla tehdään. Toiminnot voivat olla esim. paketin uudelleen ohjaus, paketin otsikoiden muokkaus, toiseen vuotauluun siirtäminen tai pudottaminen. Time-out-kenttä kertoo ajan, jonka jälkeen vuo poistuu taulusta, mikäli sen sääntöihin ei tule osumia. Cookie-arvo on kontrollerin määrittelemä arvo, jonka perusteella kontrolleri pystyy filtteröimään paketteja. Flags-kenttä vaikuttaa siihen, miten vuomerkintää hallitaan. OF-kytkimen konfiguroinnin kannalta oleellisimmat kentät vuotaulussa ovat Match Fields ja Instructions (Appenzeller, ym. 2014, 25.) Näiden kenttien toimintaa kuvataan kuviossa 6.

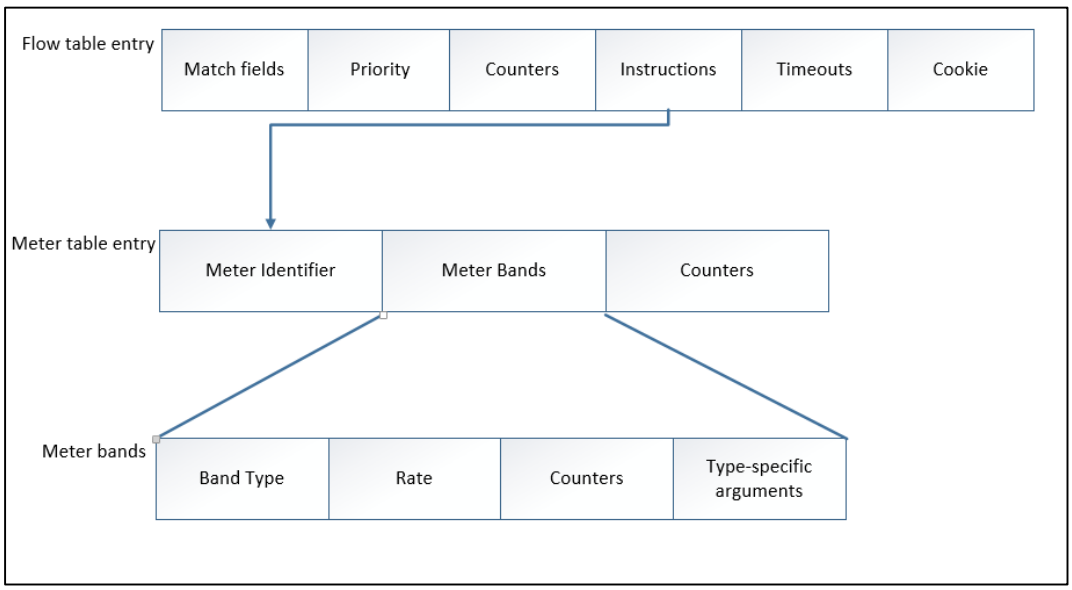


Kuvio 6. OF-kytkimen vuotaulun toiminta (Appenzeller ym. 2014, 25)

Saapuvan paketin osalta tarkastellaan aluksi otsikko tietoja, jonka perusteella paketti yhdistetään yhteen tai useampaan vuotietoon. Vuon perusteella kytkimellä on ohjeet mitä paketille pitää tehdä, jonka perusteella sille tehdään mahdollisesti paketin muokkausta ja sen jälkeen ohjataan joko uuteen vuotauluun tai ulos tietystä kytkinportista (Appenzeller, ym. 2014, 25.)

3.4 OpenFlow-protokollan palvelunlaatutuki

OpenFlow tarjoaa kaksi työkalua palvelun laadun toteuttamiseen SDN-verkossa. OF-protokolla ei itsessään mahdollista jonojen konfigurointia OF-kytkimen fyysiseen rajapintaan, mutta se mahdollistaa jonojen sitomisen vuosääntöihin. OF käyttää versiosta 1.3 eteenpäin set-queue käskyä paketin liittämiseen tiettyyn jonoon. Toinen palvelun laatutyökalu OFissa on meter-konfiguraatiot. Meter on kytkimen elementti, joka mittaa ja hallinnoi nopeutta, jolla paketteja ohjataan rajapinnasta ulos. Mikäli tiedonsiirto nopeus ylittää tai alittaa meteriin määritellyn rajan, meter aktivoi bandin. Band määrittelee toiminnot, jolla tilanne pyritään normalisoimaan. Tämä toiminto voi olla joko pakettien pudotus tai uudelleen merkkaus alempaan prioriteettiluokkaan. Meter voidaan sitoa yhteen tai useampaan vuosääntöön (Stallings 2015, 297-299.) Meterin rakenne OF-kehyksessä näkyy kuviossa 7.



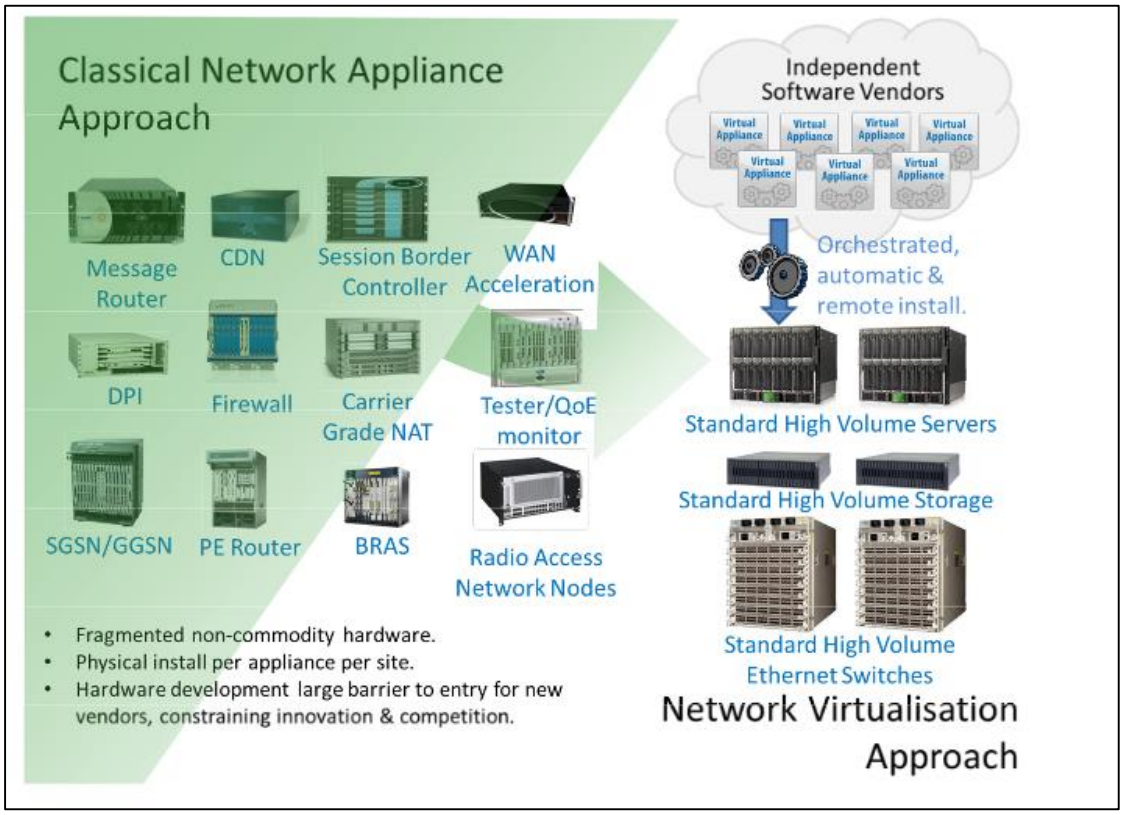
Kuvio 7. OpenFlow meter (Stallings 2015, 299)

Meter kenttä löytyy OF-kehyksen Instructions-kentän sisältä. Meter-kenttä koostuu Identifierista, jolla meter tunnistetaan. Meter Bands sisältää meterin konfiguraatiot. Band Type kertoo mitä raja-arvojen ylittyessä tehdään, eli tiputetaanko paketit vai muutetaanko niiden merkkausta. Rate määrittelee ylä – ja alarajan sallitulle tiedonsiirtonopeudelle. Counters laskee sisään tulevien pakettien määrää (Stallings 2015, 297–299.)

4 Network Function Virtualization

4.1 Yleistä

NFV pyrkii muuttamaan verkko-operaattoreiden verkkoarkkitehtuuria virtualisoinnin avulla. Suuri osa verkkolaitteista voitaisiin korvata tehokkailla palvelimilla, joissa pyörivät virtuaalikoneet toimivat verkkokomponentteina kuten reitittiminä, kytkiminä tai palomuureina. NFV ja SDN tukevat vahvasti toisiaan samankaltaisten arkkitehtuurillisten periaatteiden myötä, mutta ne eivät ole riippuvaisia toisistaan. SDN:n tarjoama välitys – ja kontrollitason eristäminen voi lisätä NFV-malliin perustuvan datakeskuksen suorituskykyä ja dynaamisuutta merkittävästi. Molemmat tekniikat pohjautuvat myös osittain ohjelmisto – ja rautakehityksen eristämiseen toisistaan (Benitez, Bugenhagen, Chiosi, Clarke, Cui, Damker, Delisle, Demaria, Deng, Fargano, Feger, Fukui, Guardini, Khan, Kolias, Lopez, Loudier, Matsuazaki, Manzalini, Michel, Minerva, Ogaki, Reid, Ruhl, Salguero, Sen, Shimano & Willis 2012, 5-6.) NFV:n ajattelumalli verrattuna perinteiseen tietoverkkoarkkitehtuuriin esitetään kuviossa 7.



Kuvio 7. NFV-malli verrattuna perinteiseen tietoverkkoon (Benitez ym. 2012, 5)

Perinteisessä tietoverkossa eri verkkoelementeille on olemassa erilaisia fyysisiä laitteita, joiden toimintaa rajoittuu tiettyyn verkon toiminallisuuteen. NFV:ssä nämä toiminallisuudet toteutetaan ohjelmistolla, jota ei ole ohjelmoitu tiettyyn rautaan, vaan se voidaan tuoda tavalliselle palvelimelle. NFV-mallissa tietoverkko koostuisi palvelimista, tietoa varastoivista laitteista ja suuren porttimäärän omaavista kytkimistä. Tällä tavoin ohjelmisto ei ole sidottuna tietyn laitevalmistajan rautaan, ja se voidaan tuoda verkkoon ilman tarvetta asentaa uusia verkkolaitteita (Benitez ym. 2012, 5.)

4.2 Hyödyt

NFV:n tarjoamat mahdollisuudet tuovat mukanaan useita etuja verkko-operaattorille. Laitekustannukset pienenevät ja myös sähkönkulutus vähenee merkittävästi, kun eri verkkolaitteita pystytään yhdistämään samalle palvelimelle. Muutosten teke-

minen verkkoon nopeutuu, koska ohjelmistojen kehitys ei ole sidottuna tiettyyn fyysiseen laitteeseen. Testiympäristöjen pystytys on helpompaa, ja ne voivat jossain määrin hyödyntää samaa infrastruktuuria tuotantoverkon kanssa. Tämä vähentää suunnitteluun ja käyttöönottoon liittyviä kustannuksia. Tarjottavia verkkopalveluita pystytään nopeasti skaalamaan asiakkaan tarpeen mukaan. Muutoksia pystytään tekemään dynaamisemmin liikennevirtojen ja palvelun tarpeen mukaan. Alalle voidaan saada nopeampaa kehitystä pienempien yritysten innovaatioiden kautta, koska ohjelmistokehityksessä on helpompi kilpailla pienemmällä budjetilla verrattuna raudan kehitykseen. Ohjelmistojen kehitys nopeutuu, koska ne eivät enää ole sidottuna laitevalmistajan rautaan (Benitez ym. 2012, 8-9.)

4.3 Haasteet

NFV:ssä on tällä hetkellä olemassa myös suuria haasteita, jotka hankaloittavat ja hidastavat tekniikoiden yleistymistä. Vaikka ohjelmistot eivät ole enää sidottuna verkkolaitteisiin, voi ohjelmistojen käyttöönotto erilaisten datakeskusten erilaisiin virtualisointialustoihin tuoda ongelmia. Perinteisten verkkolaitteiden rauta ja käyttöjärjestelmä on optimoitu aina yhtä toiminnallisuutta varten. Tavallisia palvelimia käytettäessä on odotettavissa laskua verkkokomponenttien suorituskyvyssä. NFV-laitteiden on toimittava yhdessä verkko-operaattoreiden tavallisten verkkolaitteiden kanssa eli NFV ja vanhan mallinen tietoverkko muodostavat ns. hybridiverkon. NFV vaatii toimiakseen vakaan hallinta-arkkitehtuurin, jossa muutoksia pystytään tekemään dynaamisesti. Verkon toimintojen on oltava automatisoitavissa, mikäli skaalautuvuuden osalta halutaan saada etua vanhan malliseen verkkoon nähden. Kaikki tämä pitäisi pystyä toteuttamaan tavalla, jossa verkon hallinta olisi yksinkertaisempaa kuin nykyisissä tietoverkoissa (Benitez ym. 2012, 10–12.)

5 Palvelun laatu

5.1 Yleistä

Nykypäivän tietoverkoissa kulkee paljon erityppistä dataa käyttäen samaa fyysistä infrastruktuuria. Osa datasta on tärkeämpää ja tarvitsee näin ollen erityyppistä kohtelua. Tärkeä data voi olla esimerkiksi verkon toiminnan kannalta tärkeitä protokollia kuten jokin reititys -tai hallintaprotokolla. Se voi myös olla reaaliaikaista liikennettä kuten ääntä tai kuvaa, joka ei kestä yhtä suurta viivettä ja joka vaati enemmän kaistaa. Tämän takia tarvitaan tietoverkkoja, joissa liikennettä voidaan luokitella tärkeysjärjestykseen, jonka perusteella verkkolaitteet käsittelevät eri liikenneluokkia eri tavoin (Arindam 2013, 2.)

Toimivat palvelunlaatuasetukset ovat osa kannattavaa liiketoimintaa. Palvelun tarjoajan on pystyttävä tarjoamaan asiakkailleen SLA-sopimus, jossa asiakkaalle luvataan verkon toimivuus sovittujen parametrien mukaan. SLA-sopimuksessa määriteltäviä parametreja on esitelty tarkemmin kappaleessa 6. SLA-sopimuksessa voidaan määritellä erilaisia saatavuuksia, kaistanleveyksiä ym. tietyille palveluille. Sopimuksen ulkopuolelle jäävä liikennettä käsitellään ruuhkatilanteissa joko pudottamalla se tai käsittelemällä se pienellä prioriteetilla (Chapman & Joseph 2009, 71-73.)

5.2 Integrated Services

IntServ-mallissa palvelun laatu taataan verkon päästä päähän käyttämällä RSVP:tä signaloimaan kaikille reitillä oleville laitteille tarvittavat tiedot tietoliikennevoista. Vuo tarkoittaa yksisuuntaista kahden sovelluksen välistä tietovirtaa. Yksittäiset vuot erotellaan toisistaan viidellä parametrilla: lähde IP-osoite, lähde portti, kohde IP-osoite, kohde portti ja käytetty protokolla. Tietovirran alkaessa RSVP viestittää palvelulaatu vaatimukset kaikille verkon laitteille. IntServ-mallissa laitteet voivat pyytä kahden tyyppistä palvelua. Palvelu, jolle on taattu pieni viive ja kaikki mahdollinen

kaistanleveys pahoissa ruuhkatilanteissa, tai palvelu, jolla on BE:tä suurempi prioriteetti ja käytössä enemmän kaistaa kevyen ja keskisuurten liikennemäärien aikana. Kaikki muu liikenne menee BE-liikenteenä eli sitä käsitellään alimmalla prioriteetilla. IntServ-mallin toimivuudessa on paljon ongelmia. Kaikkien verkkolaitteiden, mukaan lukien päätelaitteet, on ymmärrettävä RSVP:tä, jotta ne osaavat pyytää palvelun laatua ja signaloida sitä eteenpäin. Koska palvelunlaatuasetukset muodostetaan vasta tietovirran alkaessa, ovat verkkolaitteiden konfiguraatiot ”pehmeitä”. Tämä tarkoittaa, että ne vaativat säännöllistä päivitystä RSVP:llä, joka tuo verkkoon lisää kuormitusta. Näiden tietojen ylläpitäminen lisää verkkolaitteiden resurssivaatimuksia. Näistä syistä IntServ skaalautuu huonosti isoihin tietoverkkoihin, joissa yhtäaikaisten tietovirtojen määrä voi olla jopa miljoonissa (Chapman & Joseph 2009, 26-27.)

5.3 Differentiated Services

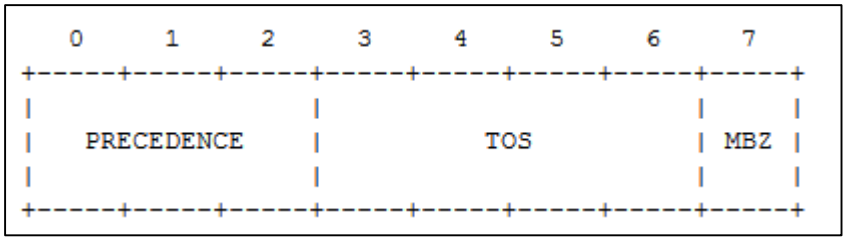
DiffServ-malli on suunniteltu tarjoamaan skaalautuvampi vaihtoehto palvelun laadun varmistamiseksi tietoverkoissa. DiffServissä jokainen verkkolaite tekee omat päätöksensä siitä, miten saapuvia paketteja käsitellään. Tätä kutsutaan termillä PHB. Jotta verkkolaite tietää, mitä palvelun laatua tietovirralle pitää asettaa, merkataan paketit IP-paketin DSCP-arvolla. Kaikkia paketteja joilla on sama DSCP arvo ja suunta, kutsutaan termillä BA. Näin ollen eri IP-osoitteista ja eri sovelluksista saapuvat paketit voivat kuulua saman BAn alle (Chapman & Joseph 2009, 27.)

5.4 Merkkaus

5.4.1 Type of Service

Palvelun laadun toteutumiseksi erityyppisten liikennevirtojen kesken on paketeille tehtävä merkkausta, jotta verkkolaitteet osaavat käsitellä niitä oikeiden sääntöjen

mukaan. Pakettien merkkaus tehdään yleensä verkon reunalaitteilla. Paketin merkkauksen jälkeen jokainen verkon laite pystyy toimimaan PHB:n mukaisesti. IP-pakettien merkkaukseen käytettiin alun perin IP-otsikossa olevaa yhden tavun mittaista ToS-kenttää (Baker, Black, Blake & Nichols 1998, 6.) ToS-kenttä esitetään kuvissa 8.



Kuvio 8. IP ToS-kenttä (Almquist 1992, 3)

ToS-kentän kolme ensimmäistä bittiä määrittelevät paketin Precedence-arvon eli prioriteetin. Kolmen bitin mittaisella kentällä voidaan määrittä 8 eri prioriteettiluokkaa. Seuraavat neljä bittiä muodostavat ToS-kentän, jotka kertovat minkä tyyppisestä palvelusta on kyse ja mitä sen vaatimukset ovat viiveen, kaistan ja luotettavuuden osalta. Viimeinen bitti ToS-kentästä on toistaiseksi käyttämätön. MBZ tarkoittaa, että tämän kentän arvon on oltava aina 0 (Almquist 1992, 3-4.) ToS-kentän eri arvot esitetään taulukossa 2.

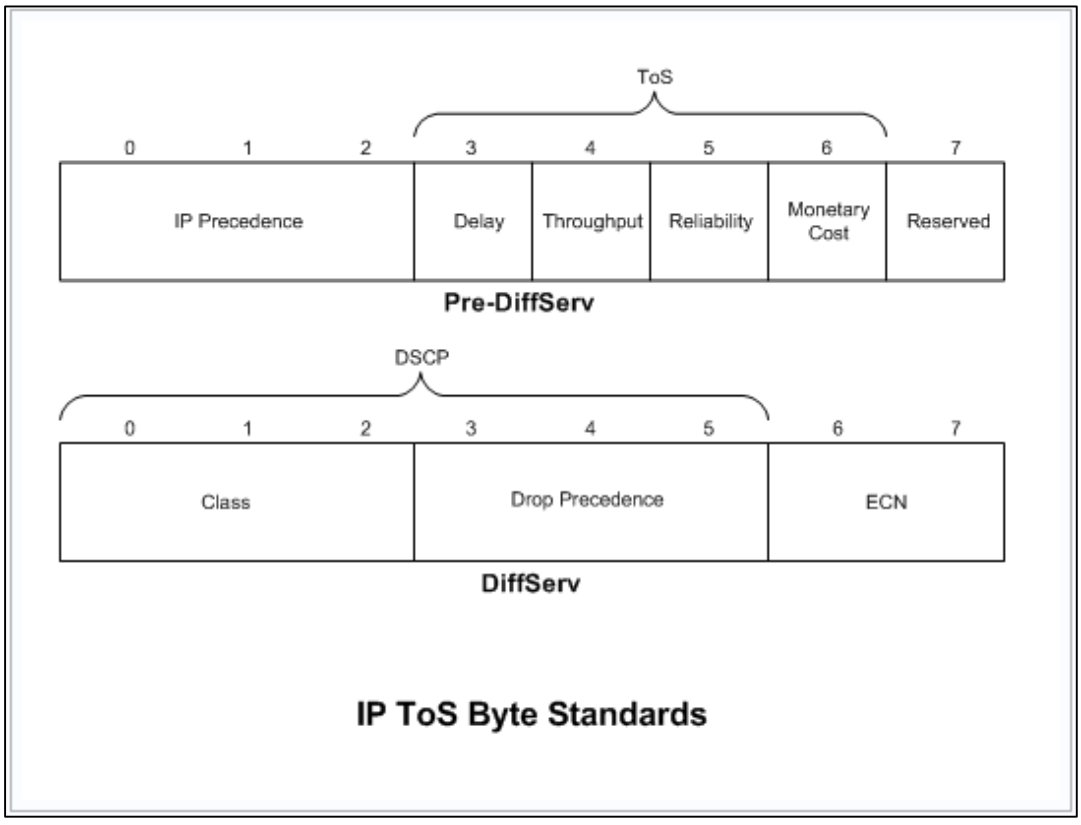
Taulukko 2. IP ToS-kentän arvot (Kumar 2012)

Bits	Fields
7-5	IP Presedence: 111 Network Control 110 Internetwork Control 101 Critic/ECP 100 Flash Override 011 Flash 010 Immediate 001 Priority

	000 Routine
4	1 = low elay, 0 = normal delay
3	1 = high throughput, 0 = normal throughput
2	1 = high reliability, 0 = normal reliability
1	1 = minimise monetary cost
0	Must be 0

5.4.2 Differentiated Services Code Point

DiffServ standardissa määritellään uusi tapa hyödyntää IP-paketin ToS-kenttää. DSCP-kenttä esitetään kuviossa 9 verraten sitä vanhan ToS-kentän käyttöön.



Kuvio 8. DSCP & ToS (Andy 2008)

Ensimmäiset kuusi bittiä muodostavat kentän DSCP-osuuden. Ensimmäiset kolme bittiä merkitsevät paketin prioriteettia samalla tavalla kuin alkuperäisessä ToS-kentässä. Tämä osa paketista on taaksepäin yhteensopiva, sillä arvoille on vain keksitty uudet nimitykset. Bitit 3-5 merkkaavat paketin pudotustodennäköisyyttä. Viimeiset kaksi bittiä muodostavat ECN-kentän, joka on toistaiseksi käyttämätön (Chapman & Joseph 2009, 24–25.) DSCP-kentän eri arvojen merkitykset esitetään taulukossa 3.

Taulukko 3. DSCP-kentän arvot (Kumar 2012)

DSCP Name	DS Field Value		IP Presedence
	Binary	Decimal	
CS0	000 000	0	0
CS1	001 000	8	1
AF11	001 001	10	1
AF12	001 010	12	1
AF13	001 110	14	1
CS2	010 000	16	2
AF21	010 001	18	2
AF22	010 010	20	2
AF23	010 110	22	2
CS3	011 000	24	3
AF31	011 001	26	3
AF32	011 010	28	3
AF33	011 110	30	3
CS4	100 000	32	4
AF41	100 001	34	4
AF42	100 010	36	4
AF43	100 110	38	4
CS5	101 000	40	5
EF	101 110	46	5
CS6	110 000	48	6

CS7	111 000	52	7

Vasemmalla taulukossa näkyy DSCP-arvon nimi. CS0-CS7 arvojen käyttö takaa yhteensopivuuden IP Precedencen kanssa, jota vastaava arvo näkyy taulukon oikeassa sarakkeessa. AF eri arvoineen tarkoittaa eri prioriteettien omaava liikennettä. EF on kaikkein korkeimman prioriteetin omaavaa liikennettä DiffServ-mallissa (Chapman & Joseph 2009, 29.)

5.5 Jonot

5.5.1 Yleistä

Palvelun laadun toteutumisen vuoksi on pystyttävä hallitsemaan verkkoon syntyviä ruuhkatilanteita. Ruuhkanhallinta koostuu kolmesta vaiheesta: Ensiksi verkkolaitteen rajapintoihin tehdään tarvittavat jonot. Toiseksi konfiguroidaan pakettien luokittelu ja merkkaus liikenneluokan – ja tyypin mukaan. Kolmanneksi paketit vuorotellaan merkkauksen mukaan, jonka perusteella määritellään, mitkä jonossa olevista paketeista käsitellään ensin. Oleellinen osa ruuhkanhallintaa on myös ruuhkautumista ennalta ehkäisevät toimenpiteet (Chapman & Joseph 2009, 45-46.)

Jonot ovat välttämätön osa ruuhkanhallintaa palvelun laatua vaativissa verkoissa. Jonoja käytetään tietoliikenteen varastointiin pakettien odottaessa prosessointia tai sarjallistamista (eng. serialization). Verkkolaitteiden rajapinnoissa on erilliset jonot sisääntulo- kuin myös ulosmenoliikenteelle. Sisääntulojono varastoi paketteja, kunnes verkkolaitteen prosessori käsittelee paketin ja ohjaa sen oikealle rajapinnalle. Ulostulojono varastoi paketteja, kunnes verkkolaite pystyy sarjallistamaan paketin fyysiseen linkkiin. Jonot ovat välttämätön osa ruuhkanhallintaa palvelun laatua vaativissa verkoissa. Ruuhkautuminen voi johtua siitä, että sisääntulorajapinnan nopeus on suurempi kuin ulosmenorajapinnan tai sisääntulorajapintoja on useampi kuin ulosmenorajapintoja. Myös verkkolaitteen riittämätön prosessointiteho voi aiheuttaa

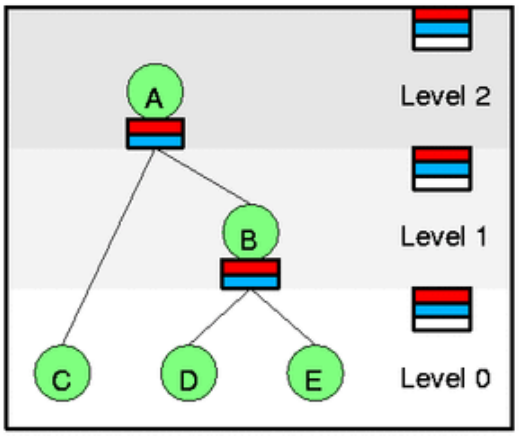
ruuhkautumista. Oletuksena kaikki liikenne ohjataan vakiojonoon, joka toimii FIFO-periaatteella (Balchunas 2010, 1-2.)

5.5.2 First in First out

FIFO on jonotyyppi, joka on oletuksena käytössä lähes kaikissa rajapinnoissa. FIFO ei vaadi erillistä konfigurointia vaan se yksinkertaisesti käsittelee ja ohjaa paketteja saapumisjärjestyksessä. FIFO ei huomioi ollenkaan pakettien prioriteettia. Kun FIFOa käyttävä jono täyttyy, uudet paketit yksinkertaisesti hylätään, kunnes jonossa on taas tilaa. Rautatasolla jonot käyttävät aina FIFOa. Mikäli halutaan käyttää hienojakoisempia jonotusalgoritmeja palvelun laadun takaamiseksi, on käytettävä ohjelmistotason jonoja. Ohjelmistotason jonotusalgoritmeissa tehdään useita jonoja, joista jokaisella on erillinen prioriteetti. Sisään saapuvat paketit ohjataan oikeaan jonon merkkauksen perusteella. Vuorottelija käsittelee korkeamman prioriteetin jonon paketit ensimmäisenä. Jokainen jono itsessään toimii kuitenkin itsessään FIFO-mallin mukaisesti (Balchunas, 2010, 3.)

5.5.3 Hierarchial Token Bucket

HTB on jonotusalgoritmi, joka on suunniteltu korvaamaan CBQ Linux-käyttöjärjestelmissä. Sekä CBQ, että HTB mahdollistavat rajapinnan kaistankäytön kontrolloimisen jakamalla fyysinen linkki useaan loogiseen linkkiin, jotka on tarkoitettu erilaisten liikennetyyppien käyttöön. HTB perustuu nimensä mukaisesti hierarkkiseen arkkitehtuuriin, jossa linkin kaistaa voidaan jakaa usealle eri tasolle (Devera 2002a.) HTB:n arkkitehtuuri esitetään kuviossa 9.



Kuvio 9. HTB-arkkitehtuuri (Devera 2002b)

Arkkitehtuurin ylin taso on ns. juuritaso, johon kuuluu fyysisen rajapinnan koko kaistanleveys. Alempi taso on lapsitaso, jolle annetaan käyttöön osa kaistasta. Lapsitasolla voi olla vielä useampi ”lehti”, joiden avulla voidaan tehdä hienojakoisempaa kaistanjakoa. Kuvion mukaisessa tapauksessa voidaan ajatella, että level 2 on fyysisen linkin koko kaistanleveys. Level 1 on yhdellä asiakkaalle annettu kaista ja Level 0 on tietyille protokollille annettu kaista (Devera 2002b.)

HTBn oleellisimmat konfiguroitavat parametrit jonoille ovat: rate, ceil, burst ja priority. Rate kertoo kaistanleveyden miniarvon, joka tietyllä liikenneluokalla pitäisi olla käytössä kaikissa tilanteissa. Rate arvo voidaan ylittää, mikäli verkossa ei ole ruuhkatilannetta. Tällöin jono lainaa kaistaa toisilta jonoilta. Ceil arvo kertoo liikenneluokan käytössä olevan kaistan maksimiarvon, joka ei ylity, vaikka fyysisellä linkillä olisi käyttämätöntä kaistaa. Ceil arvon yli menevä liikenne pudotetaan rajapinnalla. Burst arvo määritellee jonon purskeprOFiilin. Kun verkossa ei ole ruuhka jonoon varautuu purskeprOFiilin verran ylimääräistä kaistaa, jonka jono pystyy hyödyntämään ruuhkatilanteen syntyessä. Priority arvolla määritellään jonojen välinen prioriteetti, joka kertoo, minkä jonon paketit käsitellään ensimmäisenä ruuhkatilanteen syntyessä (Devera 2002a.)

6 Palvelun laadun mittaaminen

6.1 Yleistä

Kuten luvussa 5.1 todettiin, hyvän palvelun laadun tarjoaminen on tärkeä osa kannattavaa liiketoimintaa. Onnistuneiden palvelunlaatuasetusten taustalla on aina tarkasti suunniteltu ja toteutettu mittaus ja monitorointi. SLA-sopimusten rikkomisten taustalla on usein epärealistiset ja monimutkaiset lupaukset verkon laadusta. Verkon suorituskykymittauksia suunniteltaessa on tärkeä tietää, mitä halutaan mitata, miten mittaus kannattaa toteuttaa, kuinka usein mittaus toteutetaan. Verkkoa suunniteltaessa olisi tiedettävä, kuinka paljon jokainen palvelu kestää viivettä ja paljonko se vaatii kaistaa. Verkon suorituskykyä mitattaessa pitäisi pystyä parempiin tuloksiin mitä SLA-sopimus vaatii, jotta palveluntarjoajalle jää riittävästi virhemarginaalia (Chapman & Joseph 2009, 187-188.)

6.2 Palvelun laadun parametrit

Saatavuudella tarkoitetaan palvelun käytössä olemista tietyllä ajanhetkellä. Saatavuudella mitataan todennäköisyyttä, sille että palvelu on saatavilla ennalta määrättynä ajanjaksona. Saatavuus on merkittävä parametri SLA:ta laadittaessa. Saatavuuden laskemiseen on monia kaavoja, mutta yksi yleisesti käytetyistä on seuraava:

$$Saatavuus = \frac{MTBF}{(MTBF + MTTR)}$$

MTBF tarkoittaa mediaaniaikaa, joka on käyttökatkojen välillä. MTTR tarkoittaa mediaaniaikaa, joka kuluu vian huomaamisesta siihen hetkeen, kun vika on korjattu ja palvelu on taas saatavilla (Cisco 2004.)

Viiveellä tarkoitetaan aikaa, joka tietoliikenteeltä kuluu lähteestä kohteeseen. Viivettä voidaan mitata joko yhden – tai kahdensuuntaisena. Tietoliikenneverkoissa vii-

veen mittayksikkönä käytetään millisekuntia. Viive on tärkeä parametri erityisesti reaaliaikasovellusten palvelunlaadun mittaamisessa. Reaaliaikasovelluksen kuten VoIP-puhelun yhdensuuntaisena maksimiviiveenä pidetään 150 millisekuntia. Paketin kokonaisviiveaika muodostuu viidestä eri asiasta. Etenemisviive aiheutuu ajasta, joka valolla menee kulkea lähtöpisteestä kohteeseen. Valokuidussa etenemisviivettä aiheutuu noin 5ms 1000 kilometriä kohti. Etenemisviive kasvaa ongelmalliseksi lähinnä mannerten välisissä yhteyksissä, joissa pitäisi pystyä käyttämään reaaliaikasovelluksia (Chapman & Joseph 2009, 189.)

Kytkentäviive tai prosessointiviive tarkoittaa aikaa, joka reitittimellä tai kytkimellä kuluu paketin vastaanottamisesta paketin laittamiseen jonoon ulospäin menevän rajapinnan vuorottajalle. Kytkentäviiveet ovat nykypäivän tehokkaissa verkkolaitteissa hyvin pieniä 10–20 mikrosekuntia pakettia kohden. Vuorotteluviive eli toisin sanoen jonotusviive on aika, joka paketilla kuluu ulospäin menevän rajapinnan jonosta rajapinnasta poistumiseen. Tämä viive aiheutuu käytössä olevasta jonotus algoritmista ja vuorottajan jonon käytöstä. Tähän vaikuttaa käytössä olevan jonon kapasiteetti ja sisään saapuvaan liikenteen määrä ja tyyppi. Jonotusviive on viiveen osa-alueista se, johon palvelun laatuasetuksilla voidaan eniten vaikuttaa. Sarjallistamisviive (Serialization) viive on aika, joka paketilla kuluu linkkivälille siirtymiseen. Se riippuu linkkinopeudesta ja paketin koostaa. Tämä viive on mitättömän pieni nykyisillä linkkinopeuksilla. Palveluviive tarkoittaa viivettä, joka aiheutuu loppukäyttäjän laitteen aiheuttamasta viiveestä, kuten prosessoinnista (Chapman & Joseph 2009, 189.)

Tietoverkoista puhuttaessa hajonnalla tarkoitetaan viiveen vaihtelua. Viiveen vaihtelua aiheuttaa erityisesti verkon ruuhkautuminen. Kun erityyppistä tietoliikennettä priorisoidaan, ei korkean prioriteetin omaavalle liikenteelle pitäisi kuitenkaan varata liian suurta osaa kaistasta, koska se aiheuttaa suurta hajontaa pienemmän prioriteetin liikenteessä. Jos linkin käyttöaste on normaalissa tilanteessa yli 70 %, pitäisi yleisesti hankkia lisää kaistanleveyttä liian pahojen ruuhkatilanteiden välttämiseksi. Hajonta lasketaan vertaamalla keskimääräistä erotusta verkon mediaani viiveeseen (Chapman & Joseph 2009, 192.)

6.3 Mittausmenetelmät

Aktiivisella mittauksella tarkoitetaan mittausta, jossa verkkoon generoidaan liikennettä, jonka käyttäytymistä verkossa tutkitaan. Aktiivisella mittauksella mitattavia parametreja ovat esim. viive, pakettihävikki ja hajonta. Aktiivisen mittauksen hyviä puoli ovat se, että sen avulla verkkoon saadaan luotua poikkeuksellisen suuria ruuhkatilanteita, joita tutkimalla saadaan tärkeää tietoa verkon toimivuudesta. Myös liikennetyypit voidaan määritellä haluamalla tavalla, mikäli halutaan esimerkiksi mitata VOIP-liikenteen viivettä ruuhkatilanteessa. Toinen hyvä puoli on se, että aktiivisen mittauksen toteutus ei vaadi täyttä pääsyä kaikkiin verkon resursseihin, kuten reitittimiin. Mittauksen suunnittelu on myös selvästi passiivista mittausta yksinkertaisempaa ja aktiivinen mittaus skaalautuu passiivista mittaamista paremmin. Aktiivisen mittauksen heikkous on se, mittauksessa generoitava liikenne vaikuttaa verkon toimintaan eikä se kerro verkon normaalista toiminnasta. Aktiivisia mittauksia ei voida suorittaa tuotantoverkossa, koska se voi sotkea ratkaisevasta verkon kriittisiä toimintoja (Hasib & Schormans, 2003.)

Passiivinen mittaaminen tarkoittaa verkon monitorointi ilman, että liikennettä luodaan tai muokataan. Passiivinen mittaaminen vaati verkon laitteiden konfigurointia, jotta ne pystyvät tunnistamaan ja käsittelemään sisään tulevaa liikennettä. Verkkolaitteiden keräämä statistiikka voidaan sitten kerätä analysoitavaksi. Passiivisella mittauksella mitattavia arvoja ovat esim. saapuvien pakettien tai bittien määrä, jonojen täyttymisaste tai eri liikennetyypit/protokollat. Passiivisen mittauksen hyvä puoli on, että se ei vaikuta verkon toimintaan ja antaa näin tarkan kuvan verkon toiminnasta. Huono puoli aktiiviseen mittaukseen verrattuna on se, että passiivisella mittauksella ei voida tutkia verkon kriittisiä pisteitä aiheuttamalla keinotekoista ruuhkatilannetta (Hasib & Schormans 2003.)

7 Käytetyt laitteet/ohjelmat ja niiden asennukset

7.1 Raspberry Pi

Raspberry Pi säätiö kehitti Raspberry Pi:n vuonna 2011 opetuskäyttöön. Raspberry Pi on pienikokoinen, halpa tietokone, jonka avulla voidaan opetella kaikkea tietokoneisiin liittyvää raudasta ohjelmointiin. Raspberry Pi:n useat USB-portit mahdollistavat sen liittämiseen useisiin eri laitteisiin kuten näppäimistöön, hiireen, kameraan. Ne mahdollistavat myös Raspberry Pi:n käyttämisen verkkolaitteena (raspberrypi n.d.a.)

Raspberry Pi:llä on mahdollistaa käyttää useita erilaisia Linux-pohjaisia käyttöjärjestelmiä. Suositeltu käyttöjärjestelmä on Debian-pohjainen Rasbian. Rasbian on optimoitu Raspberry Pi:n raudalle, joka takaa paremman suorituskyvyn. Rasbianin mukana tulee yli 35000 Debian pakettia. Rasbian käyttöjärjestelmän kehitys on jatkuvasti käynnissä oleva yhteisöprojekti, jonka pääpainona on parantaa mahdollisimman monen Debian paketin vakautta ja suorituskykyä (raspberrypi n.d.b.)

7.2 Open vSwitch

OVS on avoimen lähdekoodin monikerroksinen virtuaalikytkin. OVS on lisensoitu avoimen lähdekoodin Apache 2 lisenssin alle. OVS on kirjoitettu alustariippumattomalla C:llä. OVS on ollut osana Linux kerneliä versioista 3.10 eteenpäin. OVS on suunniteltu toimimaan useiden palvelimien virtuaaliympäristöissä useilla eri virtualisointialustoilla. OVS tukee perinteisiä hallintarajapintoja ja toimii myös OF-kytkimenä SDN-ympäristöissä (openvswitch n.d.)

OVS:n arkkitehtuurin tärkeimpiä komponentteja ovat seuraavat: ovs-vswitchd on palveluprosessi, joka mahdollista vuopohjaisen kytkemisen ja implementoi kytkimen yhdessä Linuxin kernel moduulin kanssa. Ovsdb-server on kevyt tietokantapalvelin, jota ovs-vswitchd käyttää konfiguraatioiden tallentamiseen. Ovs-dpctl on työkalu kytkimen kernel moduulin konfigurointiin. Ovs-vsctl on ohjelma, jota käytetään OVS:n

konfigurointiin. Ovs-appctl on ohjelma, joka lähettää komentoja käynnissä oleville OVS palveluprosesseille. Ovs-OFctl on ohjelma, jolla kontrolloidaan OVS-kytkinten vuotauluja. Sen avulla kytkinten vuotauluja voidaan konfiguroida joko paikallisesti tai SDN-kontrollerin avulla (openvswitch n.d.)

Tässä työssä OVS asennettiin Raspberry Pi Model 2:lle. Raspberry Pi:llä oli käytössä Rasbian Jessie käyttöjärjestelmä kernelin versiolla 4.1.13-v7+. Asennus aloitettiin pakettien päivityksellä ja tarvittavien pakettien asennuksella.

```
root@raspberrypi-3:/home/pi# apt-get update
root@raspberrypi-3:/home/pi# apt-get install -y autoconf
libtool openssl pkg-config make gcc libssl-dev
```

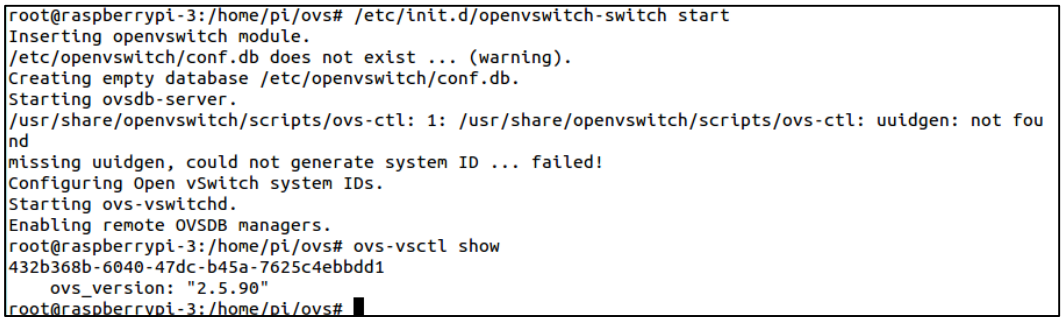
Koska sparse-ohjelmaa ei löydy Rasbianin repositioista, jouduttiin se asentamaan kopioimalla lähdekoodista.

```
root@raspberrypi-3:/home/pi# git clone git://git.ker-
nel.org/pub/scm/devel/sparse/sparse.git
root@raspberrypi-3:/home/pi# cd sparse
root@raspberrypi-3:/home/pi/sparse# git checkout -b stable
v0.4.4
root@raspberrypi-3:/home/pi/sparse# make
root@raspberrypi-3:/home/pi/sparse# make install
root@raspberrypi-3:/home/pi/sparse# cd ..
```

Tämän jälkeen OVS asennettiin kloonaamalla lähdekoodista. Tässä työssä OVS:stä käytettiin versiota 2.5.90. OVS:lle määriteltiin kansiot, joihin eri konfiguraatiotiedostot asennetaan ja tämän jälkeen OVS asennettiin ja sen käynnistysskripti kopiotiin toiseen kansioon.

```
root@raspberrypi-3:/home/pi# git clone https://github.com/open-
vswitch/ovs.git
root@raspberrypi-3:/home/pi# cd ovs
root@raspberrypi-3:/home/pi/ovs# ./boot.sh
root@raspberrypi-3:/home/pi/ovs# ./configure --prefix=/usr --
localstatedir=/var  --sysconfdir=/etc --enable-ssl
root@raspberrypi-3:/home/pi/ovs# make –j3
root@raspberrypi-3:/home/pi/ovs# make install
root@raspberrypi-3:/home/pi/ovs# cp debian/openvswitch-
switch.init /etc/init.d/openvswitch-switch
```

Asennuksen jälkeen OVS käynnistettiin kuviossa 10 näkyvällä komennolla.



Kuvio 10. OVS-käynnistys

Tulosteessa näkyy OVS-prosessin käynnistyminen ja koska kyseessä on ensimmäinen kerta, kun OVS käynnistettiin, luotiin OVSDB käynnistyksen yhteydessä.

7.3 RYU

RYU on avoimen lähdekoodin pythonilla ohjelmoitu komponenttipohjainen SDN-kontrolleri. RYU tarjoaa ohjelmistokomponentteja, joilla on määritellyt API-rajapinnat helpottamaan uusien verkon hallinta ja kontrollointi sovellusten kehitystä. Tästä syystä RYUn kehittäjät kutsuvat sitä SDN-kehykseksi, jonka päälle voidaan kehittää omia sovelluksia. RYU tukee useita verkon hallintaprotokollia kuten OF, Netconf, OF-config (What’s Ryu? n.d.)

RYU on mahdollista asentaa joko python-paketeista tai suoraan lähdekoodista. Tässä työssä RYU asennettiin python-paketteina Debian 8.2 käyttöjärjestelmän päälle. Asennus aloitettiin pakettien päivityksellä.

```
root@ryu:~# apt-get update
```

Koska RYU asennetaan python-paketeista, on asennettava pip-työkalu python-pakettien hallinnointiin.

```
root@ryu:~# apt-get install python-pip
root@ryu:~# apt-get install python-dev libxslt-dev
```

Tämän jälkeen pipin avulla asennettiin tarvittavat python-paketit, jotta kaikki RYUn toiminnallisuudet saatiin käyttöön, ja Ryu voitiin asentaa.

```
root@ryu:~# pip install eventlet
root@ryu:~# pip install pyroute2
root@ryu:~# pip install webob
root@ryu:~# pip install paramiko
root@ryu:~# pip install https://pypi.python.org/pack-
ages/source/s/six/six-1.9.0.tar.gz
root@ryu:~# pip install ryu
```

RYU asentuu oletuksena kansioon /usr/local/lib/python2.7/dist-packages.

7.4 VyOS

VyOS on Vyattan päälle kehitetty avoimen lähdekoodin Linux-pohjainen käyttöjärjestelmä, joka tarjoaa useita verkkolaitteiden ominaisuuksia kuten reitityksen useilla eri protokollilla, palomuurin ja VPN-yhteydet. VyOSissa on rautapohjaisia reitittimiä muistuttava komentolinja. VyOSissa on tilallinen konfiguraatiojärjestelmä, joka mahdollistaa useiden muutosten hyväksymisen tai hylkäämisen yhdellä komennolla ja vanhoihin konfiguraatioihin palaamisen. VyOS toimii useilla erilaisilla fyysisillä ja virtualisoiduilla alustoilla (VyOS 2016.)

VyOS asennettiin erilliselle virtuaalikoneelle levykuva-asennuksena. VyOSin asennus ei vaadi muuta kuin levykuvan asettamisen virtuaalikoneelle, jonka jälkeen asennus etenee liittessä 2 esitetyllä tavalla. VyOSin oletuskäyttäjä ja salasana ovat molemmat vyos.

7.5 Open Networking Operating System

ONOS on verkko-operaattoreille ja palveluntarjoajille tarkoitettu avoimen lähdekoodin SDN-kontrolleri. ONOS-projektin takana on useita suuria yrityksiä kuten AT&T, NTT communications, SK Telecom, Cisco, Fujitsu, Huawei ja Intel. ONOSin on tarkoitus tarjota korkeatasoista skaalautuvuutta, saatavuutta ja suorituskykyä samalla tarjoten alustan, jonka päälle on helppo rakentaa uusia applikaatioita ja palveluita. ONOS toimii klusteripohjaisena käyttöjärjestelmänä, jolloin se skaalautuu suurikokoisiin tietoverkkoihin (onosproject n.d.)

ONOSia käytettiin tässä työssä SDN-testbedin laajennuksen testauksessa, koska ONOS oli eniten testattu SDN-kontrolleri testiympäristössä. Tässä työssä ONOSissa käytettiin versiota 1.4. ONOSia ajettiin Debian 8.2-käyttöjärjestelmän päällä. ONOSin asennus aloitettiin lataamalla tarvittavia ohjemistoja kuten maven ja karaf.

```
onos@debian:~# sudo apt-get update
onos@debian:~# sudo apt-get install git-core wget maven
onos@debian:~# mkdir Downloads Application
onos@debian:~# cd Downloads
onos@debian:~/Downloads# wget
http://achive.apache.org/dist/karaf/3.0.3/apache-karaf-
3.0.3.tar.gz
onos@debian:~/Downloads# wget
http://archive.apache.org/dist/maven/maven-3/3.3.9/bin-
ries/apache-maven-3.3.9-bin.tar.gz

onos@debian:~/Downloads # tar -zxvf apache-karaf-3.0.3.tar.gz -
C../Applicatins/
onos@debian:~ /Downloads # tar -zxvf apache-maven-3.3.9-
bin.tar.gz -C../Applications/
onos@debian:~# sudo apt-get install software-properties-common
-y
onos@debian:~# sudo add-apt-repository ppa:webupd8team/java -y
onos@debian:~# sudo apt-get update
```

Koska ONOS on Java-pohjainen ohjelmisto, piti Debianille asentaa java8.

```
onos@debian:~# echo "deb
http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" |
tee /etc/apt/sources.list.d/webupd8team-java.list
onos@debian:~# echo "deb-src
http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" |
tee -a /etc/apt/sources.list.d/webupd8team-java.list
onos@debian:~# apt-key adv --keyserver hkp://keyserver.ub-
untu.com:80 --recv-keys EEA14886
onos@debian:~# sudo apt-get update
```

```
onos@debian:~# sudo apt-get install oracle-java8-installer ora-
cle-java8-set-default –y
```

Javan asennuksen jälkeen kloonattiin ONOSin lähdekoodi ja valitaan haluttu ONOS-versio. Tässä työssä käytettiin versiota 1.4, koska se oli todettu toimivaksi ja vakaaksi.

```
onos@debian:~# git clone https://gerrit.onosproject.org/onos
onos@debian:~# git checkout onos-1.4
```

Seuraavaksi määriteltiin ONOS käyttäjän tiedot muokkaamalla ~/.bashrc tiedostoa. Tiedostoon lisättiin seuraavat kolme riviä.

```
. ~/onos/tools/dev/bash_profile

export ONOS_USER=onos
export ONOS_GROUP=onos
```

Tämän jälkeen ONOS asennettiin seuraavilla komennoilla. ONOSista käytettiin versiota 1.4, koska se oli todettu toimivaksi ja vakaaksi.

```
onos@debian:~# . ~/.profile
onos@debian:~# cd onos
onos@debian:~/onos# git checkout onos-1.4
onos@debian:~/onos# mvn clean install
```

Kun ONOS oli asennettu, muutettiin tietokoneen kaapelointia niin, että ONOSilla ei enää ollut internet-yhteyttä vaan se oli osa SpiderNet-ympäristöä. ONOSille vaihdettiin IP-osoitteeksi 172.16.0.10/24. Tämän jälkeen muokattiin ~/onos/tools/test/cells/local tiedostoa lisäämällä siihen kaksi riviä.

```
export OC1=172.16.0.10
export ONOS_APPS="drivers,OF"
```

Tiedostossa määritellään IP-osoite, johon ONOS paketti asennetaan ja applikaatiot, jotka käynnistetään ONOSin käynnistyksen yhteydessä. Tämän jälkeen ajetaan cell komento äsken luodulle tiedostolle ja konfiguroidaan etäyhteys mahdollisuus ONOSin komentolinjalle.

```
onos@debian:~/onos# cell local
onos@debian:~/onos# onos-push-keys 172.16.0.10
onos@debian:~/onos# onos-install –f $OC1
```

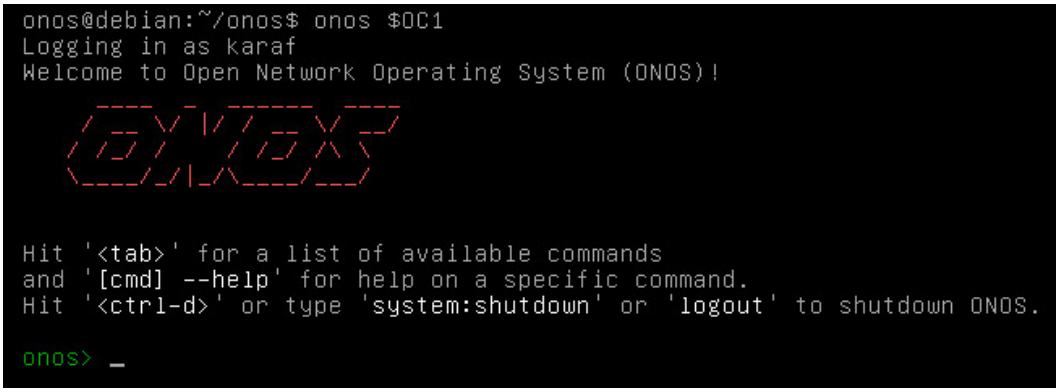
Tämän jälkeen tehtiin onos.service niminen tiedosto kansioon /etc/systemd/system. Tiedostoon lisättiin seuraavat rivit:

```
[Unit]
Description=onos SDN controller
After=network.target auditd.service
[Service]
ExecStart=/opt/onos/bin/onos-service
User=root
[Install]
WantedBy=multi-user.target
```

Seuraavaksi ONOS käynnistettiin, ja asetettiin käynnistymään bootin yhteydessä.

```
sudo systemctl start onos
sudo systemctl enable onos
```

Tämän jälkeen ONOS oli käyttövalmis ja sen komentoriville päästiin komennolla ”onos $OC1”. ONOSin käyttöliittymään kirjautuminen esitetään kuviossa 11.



Kuvio 11. ONOS CLI

7.6 JDSU MTS-6000A

JDSU MTS-6000A on tietoverkkojen testaukseen tarkoitettu tietoliikennegeneraattori. Laitteen yli 40 erilaista moduulia mahdollistavat monipuoliset testausmahdollisuudet erityyppisille verkoille. MTS-6000A pystyy generoimaan liikennettä jopa 10Gbit/s. Generoitu liikenne voidaan jakaa 10 eri liikennevirtaan, joille voidaan määrittä erilliset parametrit mittausten monipuolistamiseksi. Laitteen kosketusnäytöllä toimiva graafinen käyttöliittymä mahdollistaa testauksen nopean konfiguroinnin ja tulosten arvioinnin laitteen kirjoittamien raporttien perusteella (JDSU 2013.)

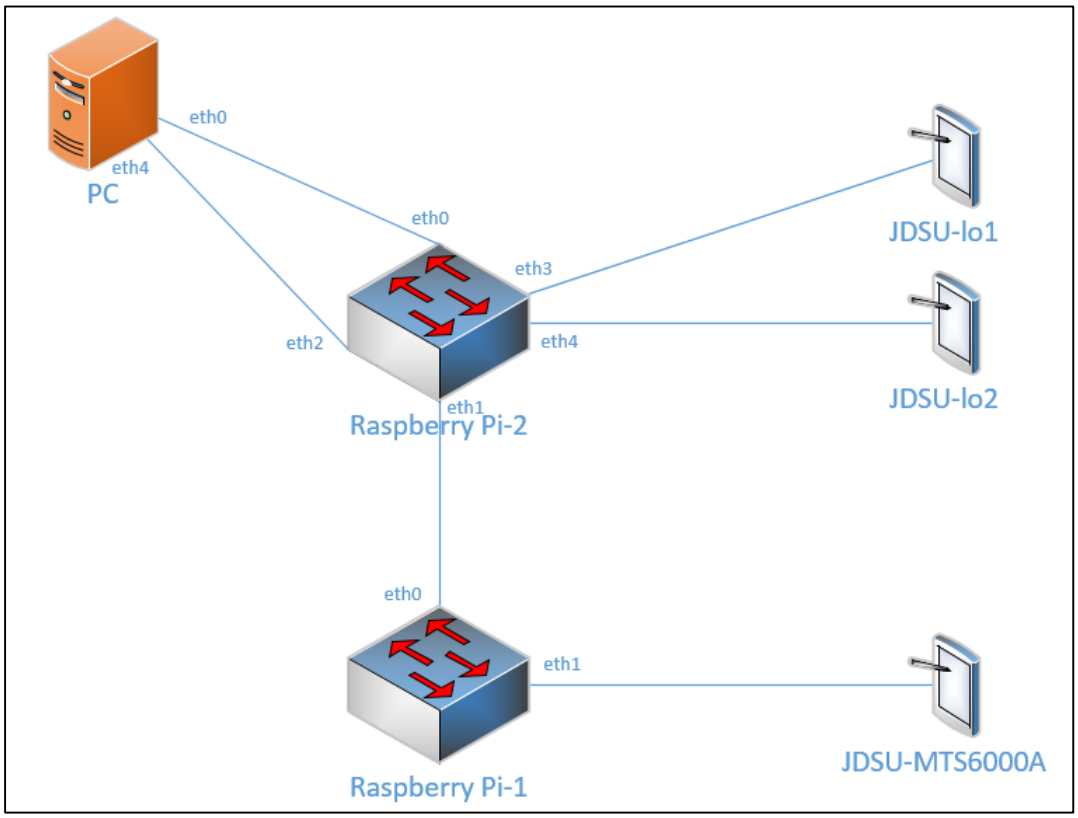
8 Toteutus

8.1 Lähtökohdat

Tutkitun teorian pohjalta haluttiin testata millä avoimen lähdekoodin SDN-ohjelmistoilla voidaan toteuttaa palvelun laatua vaativia verkkoja. OF-kytkimeksi työssä valittiin OVS, koska siitä on tullut yleisimmin käytössä oleva virtuaalinen SDN-kytkin. Työn eri vaiheissa testattiin lyhyesti neljää eri SDN-kontrolleria: Opendaylightia, Floodlightia, ONOSia ja RYU:ta. RYU päätyi SDN-kontrolleriksi lopulliseen toteutukseen, koska sen rest_qos-applikaatioilla vaadittavat konfiguraatiot (liikenteen merkkaus ja jonoihin ohjaus) onnistuivat vaivattomimmin. Muista kontrollereista saattoi joko puuttua tarvittavia ominaisuuksia kokonaan tai niiden applikaatioiden dokumentointi oli niin heikolla tasolla, tai applikaatioissa oli sellaisia bugeja, että ne eivät soveltuneet tähän työhön. Mittausympäristön pystyttämisen jälkeen, verkkoon generoitiin liikennettä JDSUn MTS6000A-liikennegeneraattorilla ja verkkoon aiheutettiin ruuhkatilanne, jonka aikana verkon käyttäytymistä testattiin erilaisilla palvelun laatuasetuksilla.

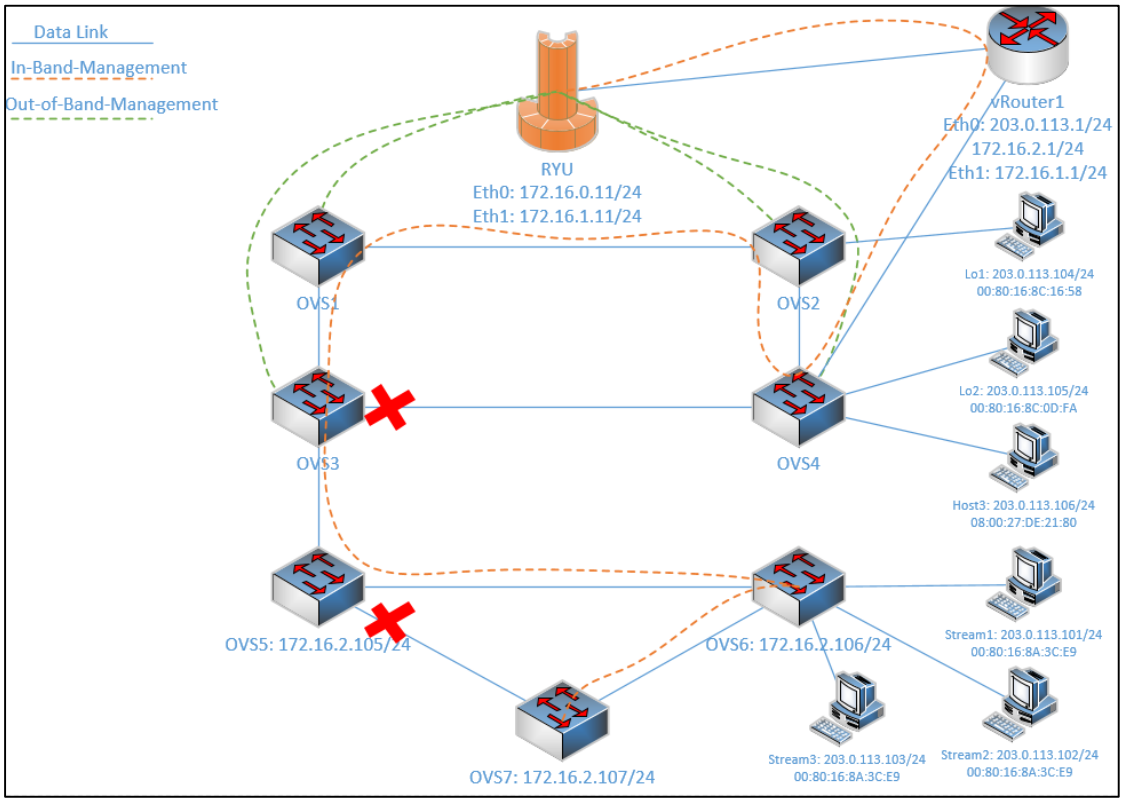
8.2 Verkkotopologiat

Palvelun laatu mittauksissa käytetyn verkon fyysinen topologia esitetään kuviossa 12.



Kuvio 12. Mittausten fyysinen topologia

Mittauksissa käytetty laitteisto koostui kahdesta Raspberry Pi Model 2:sta, yhdestä pöytäkoneesta, JDSU:n liikennegeneraattorista ja kahdesta peilaavasta mittauslaitteesta. Raspberry Piitä käytettiin OVS-kytkinten pyörittämiseen, PC:llä toimivat verkon RYU-kontrolleri, VyOS-reititin ja yksi Ubuntu desktop-kone erillisinä virtuaalikoneina. Raspberry Piiden verkkokorttien määrää lisättiin käyttämällä USB-ethernet adaptereita. Mittauksissa käytetyn verkon looginen topologia esitetään kuviossa 13.



Kuvio 13. Mittausten looginen topologia

Loogisesti verkko koostuu seitsemästä OVS-kytkimestä, RYU-kontrollerista, VyOS-reitittimestä sekä kolmesta lähettävästä ja vastaanottavasta laitteesta. OVS:t 5-7 pyörivät Raspberrypi-1:llä ja OVS:t 1-4 Raspberrypi-2:lla. Yhdelle Raspberry Piille asennettujen kytkinten väliset rajapinnat on tehty veth-virtuaalirajapinnoilla. Liikennevirrat 1-3 lähtevät kaikki JDSU-liikennegeneraattorilta. Liikennettä vastaanottavina laitteina toimivat JDSUn kaksi peilaavaa mittalaitetta sekä Host3, joka pyörii erillisellä virtuaalikoneella. Sinisellä viivalla piirretyt linkit ovat datalinkkejä, joita pitkin asiakasliikenne kulkee. Vihreä katkoviiva on OF-liikennettä varten erillinen linkki, jota pitkin OVS:t 1-4 ovat yhteydessä kontrolleriin. Oranssilla katkoviivalla merkitään ns. In-Band-Management liikennettä eli hallintaliikennettä, joka kulkee datalinkkiä pitkin, koska kaikilla verkkolaitteilla ei ole suoraa yhteyttä kontrolleriin. In-Band-Management liikenne ohjataan kontrollerille reitittimen kautta. Kaksi linkkiväliä on merkattu punaisella rastilla, koska niiden rajapinnat on suljettu manuaalisesti kytkinsilmukoiden estämiseksi.

8.3 Mittausympäristön pystyttäminen

Testiympäristön pystyttäminen aloitettiin kaapeloimalla laitteet kuvion 12 esittämällä tavalla. Tämän jälkeen kaikkien laitteiden rajapintojen IP-osoitteet konfiguroitiin kuvion 13 dokumentaation mukaisesti. RYUn ja Host3:n rajapintakonfiguraatiot tehtiin /etc/network/interfaces-tiedostoon kuvion 13 mukaisesti. Tämän lisäksi RYUlle konfiguroitiin staattinen reitti, jotta se tietää Raspberrypi1:n kytkinten löytyvän vRouter1:n kautta.

```
root@ryu~# ip route add 172.16.2.0/24 via 172.16.1.1
```

Seuraavaksi IP-osoitteet konfiguroitiin vRouter1:lle. VyOSia konfiguroitaessa on ensin siirryttävä käyttöliittymän hierarkiassa configure-tasolle, jonka jälkeen kirjoitetaan tarvittavat komennot ja ne hyväksytään commit-komennolla.

```
vyos@vyos# configure
vyos@vyos# set interfaces ethernet eth0 address 203.0.113.1/24
vyos@vyos# set interfaces ethernet eth0 address 172.16.2.1/24
vyos@vyos# set interfaces ethernet eth1 address 172.16.1.1/24
vyos@vyos# commit
```

Tämän jälkeen luotiin kytkinverkko kahdelle Raspberry Piille. Ensimmäisenä luotiin virtuaalikytkimet raspberrypi1:lle ja fyysiset rajapinnat liitettiin näihin kytkimiin.

```
root@raspberrypi-1:/home/pi# ovs-vsctl add-br ovs5
root@raspberrypi-1:/home/pi# ovs-vsctl add-br ovs6
root@raspberrypi-1:/home/pi# ovs-vsctl add-br ovs7
root@raspberrypi-1:/home/pi# ovs-vsctl add-port ovs5 eth0
root@raspberrypi-1:/home/pi# ovs-vsctl add-port ovs6 eth1
```

Tämän jälkeen tehtiin virtuaaliset rajapinnat kytkinten välille ja nekin liitettiin OVS-kytkimiin.

```
root@raspberrypi-1:/home/pi# ip link add veth1 type veth peer
name veth2
root@raspberrypi-1:/home/pi# ip link add veth3 type veth peer
name veth4
root@raspberrypi-1:/home/pi# ip link add veth5 type veth peer
name veth6
```

```
root@raspberrypi-1:/home/pi# ovs-vsctl add-port ovs5 veth1
root@raspberrypi-1:/home/pi# ovs-vsctl add-port ovs5 veth6
root@raspberrypi-1:/home/pi# ovs-vsctl add-port ovs6 veth2
root@raspberrypi-1:/home/pi# ovs-vsctl add-port ovs6 veth3
root@raspberrypi-1:/home/pi# ovs-vsctl add-port ovs7 veth4
root@raspberrypi-1:/home/pi# ovs-vsctl add-port ovs7 veth5
```

Tämän jälkeen ovs-kytkimille annettiin IP-osoitteet ja konfiguroitiin staattinen reitti, jotta ne osaavat lähettää kontrollerille menevän liikenteen vRouter1:n kautta.

```
root@raspberrypi-1:/home/pi# ifconfig ovs5 172.16.2.105/24
root@raspberrypi-1:/home/pi# ifconfig ovs6 172.16.2.106/24
root@raspberrypi-1:/home/pi# ifconfig ovs7 172.16.2.107/24
root@raspberrypi-1:/home/pi# ip route add 172.16.1.0/24 via
172.16.2.1
```

Tämän jälkeen OVS:t konfiguroitiin ottamaan TCP-yhteys RYU-kontrolleriin. Yhteyden muodostamista varten pitää määritellä kytkin, kontrollerin IP-osoite ja TCP-portti, johon yhteys muodostetaan. Ennen kontrolleriyhteyden muodostamista, on kuitenkin vaihdettava OVS käyttämään OF-versiota 1.3, koska käytössä olevan RYUn version applikaatiot on suunniteltu sen mukaisesti.

```
root@raspberrypi-1:/home/pi# ovs-vsctl set bridge ovs5 proto-
cols=OpenFlow13
root@raspberrypi-1:/home/pi# ovs-vsctl set bridge ovs6 proto-
cols=OpenFlow13
root@raspberrypi-1:/home/pi# ovs-vsctl set bridge ovs7 proto-
cols=OpenFlow13
root@raspberrypi-1:/home/pi# ovs-vsctl set-controller ovs5
tcp:172.16.1.11:6633
root@raspberrypi-1:/home/pi# ovs-vsctl set-controller ovs6
tcp:172.16.1.11:6633
root@raspberrypi-1:/home/pi# ovs-vsctl set-controller ovs7
tcp:172.16.1.11:6633
```

Vastaavat vaiheet tehtiin myös raspberrypi2:lle, jonka konfiguraatiot löytyvät liitteestä 2. OVS-kytkimen portit ja niiden numerointi voidaan tarkistaa ovs-ofctl-komennolla, jonka tuloste esitetään kuviossa 14. Samalla komennolla nähdään myös kytkimen datapath-id, joka toimii kytkimen tunnisteena.

```
root@raspberrypi-1:/home/pi# ovs-ofctl -O openflow13 show ovs6
OFPT_FEATURES_REPLY (OF1.3) (xid=0x2): dpid:000060e3271f2ef7
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS QUEUE_STATS
OFPST_PORT_DESC reply (OF1.3) (xid=0x3):
 1(eth1): addr:60:e3:27:1f:2e:f7
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 2(veth2): addr:8e:cb:94:16:fd:bd
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(veth3): addr:3e:42:01:b8:4a:10
     config:     0
     state:      LINK_DOWN
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(ovs6): addr:60:e3:27:1f:2e:f7
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.3) (xid=0x5): frags=normal miss_send_len=0
root@raspberrypi-1:/home/pi#
```

Kuvio 14. ovs-ofctl-tuloste

Tässä vaiheessa siirryttiin RYUn ja sen applikaatioiden käynnistykseen. Ennen RYU käynnistystä tehtiin pieni muokkaus simple_switch_13 applikaatioon, joka on applikaatio, jolla OF-kytkimet saadaan toimimaan oppivana L2-kytkimenä.

```
root@ryu~# sed '/OFPFlowMod(/,/)/s/)/, table_id=1)/'
ryu/ryu/app/simple_switch_13.py > ryu/ryu/app/qos_sim-
ple_switch_13.py
```

Kyseisellä komennolla vaihdetaan simple_switch_13-applikaatio käyttämään vuotaulua 1 taulun 0 sijasta. Tämä muutos tehdään, koska palvelun laatuasetuksiin käytettävä rest_qos-applikaatio käyttää vuotaulua 0. Uudelle muokatulle applikaatiolla annettiin nimeksi qos_simple_switch_13. Muokkauksen jälkeen käynnistettiin kaikki työssä tarvittavat applikaatiot ryu-manager-komennolla. Kaikki RYUn applikaatiot löytyvät kansiosta /usr/local/lib/python2.7/dist-packages/ryu/app.

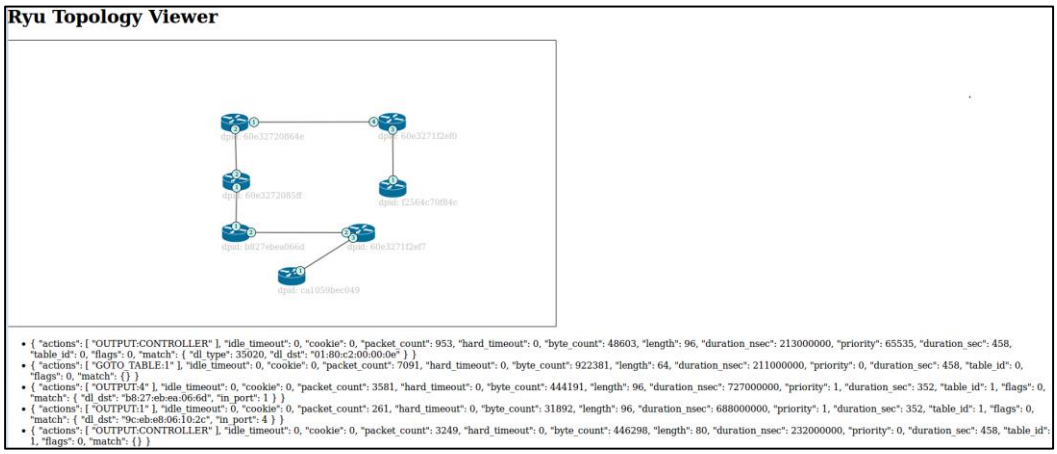
```
root@ryu~# ryu-manager --observe-links ryu.app.rest_topology
ryu.app.ws_topology ryu.app.gui_topology.gui_topology
ryu.app.rest_qos ryu.app.qos.simple_switch_13
ryu.app.rest_conf_switch
```

Kolme ensimmäistä applikaatiota käynnistävät RYUn graafisen käyttöliittymän. Rest_qos-applikaation avulla voidaan tehdä palvelun laatua koskevat asetukset, kuten IP-pakettien merkkaus ja liikenteen ohjaus jonoihin. Rest_conf_switch-applikaatio mahdollistaa kytkinkonfiguraatioiden hakemisen ja muokkauksen kontrollerilta käsin. Applikaatioiden käynnistyksen jälkeen OVS:t saivat muodostettua OF-yhteyden kontrolleriin. Kuviossa 15 näkyy RYUn onnistunut käynnistys ja kuinka kytkimet ovat muodostaneet yhteyden RYUn kanssa.

```
root@ryu:/usr/local/lib/python2.7/dist-packages/ryu/app# ryu-manager --observe-links ryu.app.rest_topology ryu.app.ws_topology ryu.app.gui_topology.gui_topology
yu.app.qos_simple_switch_13 ryu.app.rest_qos ryu.app.ofctl_rest
loading app ryu.app.rest_topology
loading app ryu.app.ws_topology
loading app ryu.app.gui_topology.gui_topology
loading app ryu.app.qos_simple_switch_13
loading app ryu.app.rest_qos
loading app ryu.app.ofctl_rest
loading app ryu.controller.ofp_handler
loading app ryu.app.rest_topology
loading app ryu.app.ws_topology
loading app ryu.controller.ofp_handler
instantiating app None of Switches
creating context switches
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app None of ConfSwitchSet
creating context conf_switch
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app ryu.app.rest_qos of RestQoSAPI
instantiating app ryu.app.ws_topology of WebSocketTopology
instantiating app ryu.app.rest_topology of TopologyAPI
instantiating app ryu.app.qos_simple_switch_13 of SimpleSwitch13
instantiating app ryu.app.ofctl_rest of RestStatsApi
instantiating app ryu.app.gui_topology.gui_topology of GUIServerApp
(11157) wsgi starting up on http://0.0.0.0:8080
/usr/local/lib/python2.7/dist-packages/ryu/topology/switches.py:550: UserWarning: Datapath#ports is kept for compatibility with the previous openflow versions (<
1.3). This not be updated by EventOFPPortStatus message. If you want to be updated, you can use 'ryu.controller.dpset' or 'ryu.topology.switches'.
  for port in dp.ports.values():
[QoS][INFO] dpid=0000b827ebea066d: Join qos switch.
[QoS][INFO] dpid=0000ca1059bec049: Join qos switch.
[QoS][INFO] dpid=000060e3271f2ef7: Join qos switch.
packet in 202481601218157 66:21:82:7e:80:2a ff:ff:ff:ff:ff:ff 1
packet in 106528730197751 66:21:82:7e:80:2a ff:ff:ff:ff:ff:ff 2
[QoS][INFO] dpid=000060e3272085ff: Join qos switch.
[QoS][INFO] dpid=000060e3272086b2: Join qos switch.
[QoS][INFO] dpid=0000f2564c70f84c: Join qos switch.
[QoS][INFO] dpid=000060e3271f2ef0: Join qos switch.
```

Kuvio 15. RYUn käynnistys

Verkon topologiakuvaa voidaan nyt tarkastella selaimelta osoitteesta http://172.16.0.11:8080. Graafisen käyttöliittymän näkymä esitetään kuviossa 16.



Kuvio 16. RYU graafinen käyttöliittymä

Kuten kuvioista 13 ja 16 nähdään, RYUn piirtämä topologiakuva vastaa verkon loogista topologiakuvaa. RYUn topologiakuvassa näkyy myös kytkinten datapath-id:t, porttien numerot, sekä kytkintä klikkaamalla saa näkyviin kytkimen vuotaulun. RYUn graafinen käyttöliittymä ei mahdollista kytkinten konfigurointia. Yhteyden muodostumisen onnistuminen voidaan tarkistaa myös kytkimeltä käsin ovs-vsctl show-komennolla. Kyseisen komennon tuloste esitetään kuviossa 17.

```
root@raspberrypi-1:/home/pi# ovs-vsctl show
3a01c1fe-e62a-43d0-b767-3b4f560b6908
    Bridge "ovs5"
        Controller "tcp:172.16.1.11:6633"
            is_connected: true
        Port "eth0"
            Interface "eth0"
        Port "veth6"
            Interface "veth6"
        Port "ovs5"
            Interface "ovs5"
                type: internal
        Port "veth1"
            Interface "veth1"
    Bridge "ovs7"
        Controller "tcp:172.16.1.11:6633"
            is_connected: true
        Port "veth4"
            Interface "veth4"
        Port "ovs7"
            Interface "ovs7"
                type: internal
        Port "veth5"
            Interface "veth5"
    Bridge "ovs6"
        Controller "tcp:172.16.1.11:6633"
            is_connected: true
        Port "eth1"
            Interface "eth1"
        Port "veth2"
            Interface "veth2"
        Port "veth3"
            Interface "veth3"
        Port "ovs6"
            Interface "ovs6"
                type: internal
    ovs_version: "2.5.90"
root@raspberrypi-1:/home/pi#
```

Kuvio 17. ovs-vsctl show-tuloste

Komento tulostaa kaikki OVS-kytkimet ja kertoo mitä portteja niihin on liitetty. Komento näyttää myös, mikäli kytkimelle on konfiguroitu TCP-yhteys kontrolleriin, ja onko yhteyden muodostus onnistunut. Yhteyden muodostumisen jälkeen RYU luo OVS:n vuotauluun kuvion 18 mukaiset vuot. Vuot voidaan tarkistaa OVS:ltä ovs-ofctl show-komennolla, johon määritellään käytössä olevan protokollaversio ja kytkin, jolta vuot halutaan tulostaa.

```
root@raspberrypi-1:/home/pi# ovs-ofctl -O openflow13 dump-flows ovs6
OFPST_FLOW reply (OF1.3) (xid=0x2):
root@raspberrypi-1:/home/pi# ovs-ofctl -O openflow13 dump-flows ovs6
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=4.269s, table=0, n_packets=4, n_bytes=204, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=4.267s, table=0, n_packets=3, n_bytes=890, priority=0 actions=goto_table:1
 cookie=0x0, duration=0.563s, table=1, n_packets=0, n_bytes=0, priority=1,in_port=2,dl_dst=b8:27:eb:ea:06:6d actions=output:2
 cookie=0x0, duration=4.284s, table=1, n_packets=3, n_bytes=890, priority=0 actions=CONTROLLER:65535
root@raspberrypi-1:/home/pi#
```

Kuvio 18. RYUn asettamat oletusvuot

Kuviossa näkyy, että oletuksena kaikki OVS:lle saapuvat paketit, jotka eivät osu yhteenkään vuohon, lähetetään kontrollerille. Tämä sääntö tehdään pienimmällä mahdollisella prioriteetillä, jolloin mikä tahansa dynaamisesti tai staattisesti luotu vuosääntö ohittaa sen vuotaulun läpikäyntijärjestyksessä.

9 Mittaukset

9.1 Mittausten valmistelu

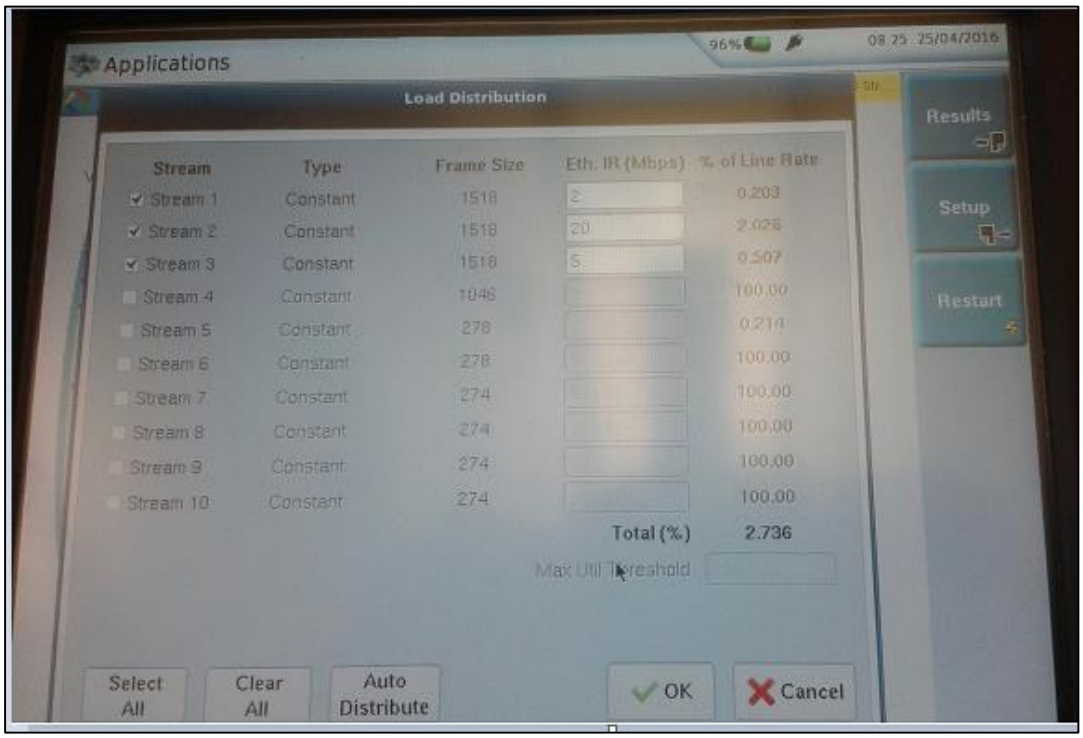
Kaikki tämän työn mittaukset toteutettiin aktiivisena mittauksena, koska testiympäristö ei ole tuotantokäytössä, eikä siellä itsessään liiku tarpeeksi liikennettä mittausten toteuttamiseksi. Ainoa liikenne, jota verkossa kulki passiivisesti syntyä OVS5-7 TCP-yhteydestä kontrollerin kanssa. Mittauksissa käytetyn verkon fyysinen ja loogi-nen topologia on käyty läpi kappaleissa 8.2.1 ja 8.2.2. Mittaukset toteutettiin käyttämällä kolmea erillistä liikennevirtaa. Tämän lisäksi verkossa kulki Raspberrypi-1:den

kytkinten kontrolliliikennettä. Mittauksissa käytetyt liikennevirrat esitetään taulukossa 4.

Taulukko 4. Mittauksissa käytetyt liikennevirrat.

Liikennevirta	IP_SRC	IP_DST	DL_DST	Nopeus	Kuvaus
Stream1	203.0.113.101/24	203.0.113.104/24	00:80:16:8C:16:5B	2mbit/s	VOIP
Stream2	203.0.113.102/24	203.0.113.105/24	00:80:16:8C:0D:FA	20mbit/s	Data1
Stream3	203.0.113.103/24	203.0.113.106/24	08:00:27:DE:21:80	5mbit/s	Data2

Kuten ylläolevasta taulukosta näkyy, liikennevirtojen yhteenlaskettu nopeus on 27mbit/s. Kaikkien mittauksien pituus oli yksi minuutti. Mittauksissa käytettävät liikennevirrat JDSUn graafisesta käyttöliittymästä katsottuna näkyvät kuviossa 19.



Kuvio 19. JDSU graafinen käyttöliittymä

Generoitavan liikenteen määrä haluttiin pitää riittävän pienenä, koska esimerkiksi Raspberry Piin rajallisten resurssien ei haluttu vaikuttavan mittaustuloksiin. Ruuhkati-

lanne saatiin muodostumaan verkkoon rajoittamalla Raspberry Piiden välisten fyysisten rajapintojen nopeudeksi 10mbit/s.

```
root@raspberrypi-1:/home/pi# ovs-vsctl set interface eth0 in-
gress_policing_rate=10000

root@raspberrypi-2:/home/pi# ovs-vsctl set interface eth1 in-
gress_policing_rate=10000
```

Ennen mittausten aloitusta testattiin verkon toiminta päästä päähän lähettämällä, jokaisesta JDSUn liikennevirrasta muutama ICMP-paketti kohdeosoitteisiin. Kun päätelaitteiden liikennettä on kulkenut verkon päästä päähän, OVS:t saavat kontrollerilta tiedon vuotauluunsa, mihin porttiin millekin laitteelle menevä liikenne kuuluu ohjata. Kuviossa 20 esitetään esimerkki OVS6:n vuotaulusta sen jälkeen, kun testilaitteilta on generoitu liikennettä verkkoon, ja RYU on lisännyt vuotaulumerkinnät OVS:n vuotauluun.

```
root@raspberrypi-1:/home/pi# ovs-ofctl -O openflow13 dump-flows ovs6
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=195.688s, table=0, n_packets=195, n_bytes=9945, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=195.686s, table=0, n_packets=63, n_bytes=14539, priority=0 actions=goto_table:1
 cookie=0x0, duration=191.982s, table=1, n_packets=0, n_bytes=0, priority=1,in_port=2,dl_dst=b8:27:eb:ea:06:6d actions=output:2
 cookie=0x0, duration=16.339s, table=1, n_packets=11, n_bytes=774, priority=1,in_port=2,dl_dst=00:80:16:8a:3c:e9 actions=output:1
 cookie=0x0, duration=16.329s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=1,dl_dst=00:80:16:8c:16:5b actions=output:2
 cookie=0x0, duration=10.018s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=1,dl_dst=08:00:27:de:21:80 actions=output:2
 cookie=0x0, duration=195.703s, table=1, n_packets=48, n_bytes=13525, priority=0 actions=CONTROLLER:65535
root@raspberrypi-1:/home/pi#
```

Kuvio 20. OVS6-dynaamiset vuot.

RYUn qos_simple_switch13-applikaatio on tehnyt OVS6:n vuotauluun numero 1 säännöt, joilla tiettyihin MAC-osoitteisiin menevä liikenne ohjataan tietystä portista ulos. Muiden kytkinten vastaavat vuotaulut löytyvät liitteistä 3-8.

9.2 Lähtöasetelma

9.2.1 Mittaus 1: Verkon toiminta oletusasetuksilla

Mittauksessa 1 haluttiin testata verkon toimintaa ilman mitään palvelun laatuasetuksia, jotta saadaan kuva verkon normaalista toiminnasta ruuhkatilanteessa. Mittauksen 1 tuloksia käytettiin pohjana vertailulle myöhempiin mittauksiin, joissa palvelun laatuasetuksilla pyritään muokkaamaan verkon käyttäytymistä. Mittauksella 1 haluttiin myös nähdä, katkeaako OVS5-7:n hallintayhteys RYUlle ruuhkatilanteissa, sen käyttäessä samaa fyysistä rajapintaa kuin asiakasdata.

9.2.2 Mittaus 2: Liikenteen leikkaus

Mittauksessa 2 testattiin maksimikaistanleveyden määrittämistä erilaisille liikennevirroille. Kaikki rajan ylittävä liikenne leikataan eli pudotetaan rajapinnalla. Mittausta varten luotiin kolme jonoa eri liikennevirroille. Ennen toista mittausta tehtiin OVS-kytkinten palvelun laatu konfiguraatiot RYUlta käsin. Vuot tehtiin RYUn rest_qos-applikaatiolla. Kyseinen applikaatio käyttää curlia vuotaulujen muokkaukseen.

```
root@ryu~# curl -X POST -d '{"priority": 100, "match": {"nw_dst":
"203.0.113.104"}, "actions":{"mark": 48}}' http://lo-
calhost:8080/qos/rules/000060e3271f2ef7

root@ryu~# curl -X POST -d '{"priority": 100, "match": {"nw_dst":
"203.0.113.105"}, "actions":{"mark": 16}}' http://lo-
calhost:8080/qos/rules/000060e3271f2ef7

root@ryu~# curl -X POST -d '{"priority": 100, "match": {"nw_dst":
"203.0.113.106"}, "actions":{"mark": 8}}' http://lo-
calhost:8080/qos/rules/000060e3271f2ef7
```

Kolme liikennevirtaa merkattiin omilla IP-DSCP-arvoilla liikenteen tunnistamiseksi. Komennoilla on määritelty vuolle prioriteetti, joka kertoo missä järjestyksessä vuotaulu käydään läpi paketin saapuessa. Match-fieldinä käytettiin liikennevirran kohteen IP-osoitetta. Tähän IP-osoitteeseen menevät paketit merkataan IP-DSCP arvolla. Kuten taulukosta 3 voidaan nähdä, käytetyt DSCP-arvot 8,16 ja 48 vastaavat IP-

Precedence arvoja 1,2 ja 6. Kuviossa 21 näkyy samat vuomerkinnät OVS6:n vuotaulussa.

```
root@raspberrypi-1:/home/pi# ovs-ofctl -O openflow13 dump-flows ovs6
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=3904.308s, table=0, n_packets=3892, n_bytes=198492, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x1, duration=3884.925s, table=0, n_packets=1, n_bytes=98, priority=100,ip,nw_dst=203.0.113.104 actions=set_field:48->ip_dscp,goto_table:1
 cookie=0x2, duration=3880.413s, table=0, n_packets=1, n_bytes=98, priority=100,ip,nw_dst=203.0.113.103 actions=set_field:16->ip_dscp,goto_table:1
 cookie=0x3, duration=3875.103s, table=0, n_packets=1, n_bytes=98, priority=100,ip,nw_dst=203.0.113.105 actions=set_field:8->ip_dscp,goto_table:1
 cookie=0x0, duration=3904.307s, table=0, n_packets=639, n_bytes=236135, priority=0 actions=goto_table:1
 cookie=0x0, duration=3796.619s, table=1, n_packets=0, n_bytes=0, priority=1,in_port=2,dl_dst=b8:27:eb:ea:06:6d actions=output:2
 cookie=0x0, duration=3711.743s, table=1, n_packets=7, n_bytes=534, priority=1,in_port=2,dl_dst=00:80:16:8a:3c:e9 actions=output:1
 cookie=0x0, duration=3711.728s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=1,dl_dst=00:80:16:8c:16:5b actions=output:2
 cookie=0x0, duration=3708.295s, table=1, n_packets=0, n_bytes=0, priority=1,in_port=1,dl_dst=00:80:16:8c:0d:fa actions=output:2
 cookie=0x0, duration=3705.081s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=1,dl_dst=08:00:27:de:21:80 actions=output:2
 cookie=0x0, duration=3904.319s, table=1, n_packets=631, n_bytes=235655, priority=0 actions=CONTROLLER:65535
root@raspberrypi-1:/home/pi#
```

Kuvio 21. OVS6-vuotaulu merkkaukset.

Kuten kuviosta näkyy, vuot on luotu vuotauluun numero 0 ja merkkauksen jälkeen paketit ohjataan vuotauluun 1, jossa paketit ohjataan ulos oikeasta portista. Kuvioissa 22 ja 23 esitetään liikennevirran 1 paketit Raspberrypi-1:den rajapinnoissa eth1 ja eth0 eli ennen ja jälkeen merkkauksen.

```
▾ Internet Protocol Version 4, Src: 203.0.113.101 (203.0.113.101), Dst: 203.0.113.104 (203.0.113.104)
    Version: 4
    Header Length: 20 bytes
  ▸ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 512
    Identification: 0x0000 (0)
  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: Unknown (254)
  ▸ Header checksum: 0xbf31 [validation disabled]
    Source: 203.0.113.101 (203.0.113.101)
    Destination: 203.0.113.104 (203.0.113.104)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▸ Data (492 bytes)
```

Kuvio 22. Liikennevirta1 ennen merkkausta

```
▾ Internet Protocol Version 4, Src: 203.0.113.101 (203.0.113.101), Dst: 203.0.113.104 (203.0.113.104)
    Version: 4
    Header Length: 20 bytes
  ▸ Differentiated Services Field: 0xc0 (DSCP 0x30: Class Selector 6; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 512
    Identification: 0x0000 (0)
  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: Unknown (254)
  ▸ Header checksum: 0xbe71 [validation disabled]
    Source: 203.0.113.101 (203.0.113.101)
    Destination: 203.0.113.104 (203.0.113.104)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▸ Data (492 bytes)
```

Kuvio 23. Liikennevirta1 merkkauksen jälkeen

Kahden muun liikennevirran vastaavat todennukset löytyvät liitteistä 9-12. Wireshark näyttää DSCP-arvon merkkauksen hexadesimaalilukuna. Kun paketit on merkattu, niiden IP-DSCP-arvoa voidaan käyttää match-fieldinä, jonka perusteella paketteja ohjataan niille kuuluviin jonoihin.

```
root@ryu~# curl -X POST -d '{"priority": 100, "match":
{"ip_dscp": "48"}, "actions":{"queue": 1}}' http://lo-
calhost:8080/qos/rules/0000b827ebea066d

root@ryu~# curl -X POST -d '{"priority": 100, "match":
{"ip_dscp": "16"}, "actions":{"queue": 2}}' http://lo-
calhost:8080/qos/rules/0000b827ebea066d

root@ryu~# curl -X POST -d '{"priority": 100, "match":
{"ip_dscp": "8"}, "actions":{"queue": 3}}' http://lo-
calhost:8080/qos/rules/0000b827ebea066d
```

Merkattua liikennettä jonoihin ohjaavat vuosäännöt tehtiin OVS5:lle, koska sen ulosmenevään rajapintaan muodostetaan ruuhkatilanne. Kuviossa 24 näkyy OVS5:n vuotaulu.

```
root@raspberrypi-1:/home/pi# ovs-ofctl -O openflow13 dump-flows ovs5
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=3972.636s, table=0, n_packets=7918, n_bytes=439449, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x1, duration=30.032s, table=0, n_packets=0, n_bytes=0, priority=100,ip,nw_tos=192 actions=set_queue:1,goto_table:1
 cookie=0x2, duration=23.449s, table=0, n_packets=0, n_bytes=0, priority=100,ip,nw_tos=64 actions=set_queue:2,goto_table:1
 cookie=0x3, duration=18.978s, table=0, n_packets=0, n_bytes=0, priority=100,ip,nw_tos=32 actions=set_queue:2,goto_table:1
 cookie=0x0, duration=3972.622s, table=0, n_packets=1053, n_bytes=286069, priority=0 actions=goto_table:1
 cookie=0x0, duration=3864.935s, table=1, n_packets=154, n_bytes=23796, priority=1,in_port=LOCAL,dl_dst=9c:eb:e8:06:10:2c actions=output:1
 cookie=0x0, duration=3864.914s, table=1, n_packets=245, n_bytes=21466, priority=1,in_port=1,dl_dst=b8:27:eb:ea:06:6d actions=LOCAL
 cookie=0x0, duration=3780.079s, table=1, n_packets=7, n_bytes=534, priority=1,in_port=1,dl_dst=00:80:16:8a:3c:e9 actions=output:2
 cookie=0x0, duration=3780.043s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=2,dl_dst=00:80:16:8c:16:5b actions=output:1
 cookie=0x0, duration=3776.605s, table=1, n_packets=1, n_bytes=60, priority=1,in_port=2,dl_dst=00:80:16:8c:0d:fa actions=output:1
 cookie=0x0, duration=3773.395s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=2,dl_dst=08:00:27:de:21:80 actions=output:1
 cookie=0x0, duration=3972.642s, table=1, n_packets=642, n_bytes=239973, priority=0 actions=CONTROLLER:65535
root@raspberrypi-1:/home/pi#
```

Kuvio 24. OVS5-vuotaulu

On huomattavaa, että vaikka RYUlta asetetussa vuosäännöissä match-fieldinä käytettiin DSCP-arvoja, OVS:n vuotauluihin ne ovat merkitty ToS-arvolla. Kun merkattu liikenne osuu vuosääntöön, se ohjataan oikeaan, jonoon jonka jälkeen siirrytään vuotauluun numero 1, jossa qos.simple_switch13-applikaatio hoitaa paketin ohjaamisen rajapinnasta ulos. Nämä vuosäännöt pysyivät samoina kaikissa mittauksissa.

Alkuperäisissä suunnitelmissa oli tehdä neljäs jono OVS5-OVS7 kontrolleriyhteydelle, mutta kontrolliliikenteen ohjaus jonoon ei ollut mahdollista, koska OVS tekee kontrolliyhteyttä varten säännön, joka yliajaa manuaalisesti konfiguroidut vuosäännöt. Tästä syystä kontrolleriyhteyden merkkausta tai jonoon ohjaamista toteuttaviin sääntöihin ei saatu osumaan yhtäkään pakettia. Toisaalta mittauksen 1 perusteella nämä säännöt eivät myöskään olleet välttämättömiä mittausten onnistumiseksi. Mittausta 2 varten tehdyt vuosäännöt pysyivät samoina kaikkien seuraavien mittausten ajan. Tämän jälkeen siirryttiin Raspberrypi-1:lle tekemään tarvittavat jonot.

```
root@raspberrypi-1:/home/pi# ovs-vsctl set port eth0
qos=@newqos -- --id=@newqos create qos type=linux-htb
queues=1=@q1,2=@q2,3=@q3 -- --id=@q1 create queue other-con-
fig:max-rate=1500000 -- --id=@q2 create queue other-config:max-
rate=4000000 -- --id=@q3 create queue other-config:max-
rate=3000000
```

Komennolla luodaan jonot q1, q2 ja q3. Jokaiselle määritellään maksimikaistanleveys yksiköllä Kbit/s. Kaikkien mittausten kohdalla jonojen numerointi vastaa liikennevirtojen numerointia. Liikennevirtojen pitäisi saada ainoastaan komennossa määrätty kaistanleveys käyttöönsä ja tämän luvun ylittävä liikenne leikataan. Tehdyt jonot voidaan tarkistaa komennolla:

```
root@raspberrypi-1:/home/pi ovs-vsctl list queue
```

9.2.3 Mittaus 3a: Taattu kaistanleveys

Mittauksessa 3 testattiin varata tietty minimikaistanleveys jokaiselle liikennevirralle ja jättää näiden varausten yli jäävä kaista vapaasti käytettäväksi. Tällöin korkealla prioriteetilla merkattu liikenne saa aina vähintään määritellyn kaistanleveyden käyttöönsä, mutta mikäli verkossa ei ole ruuhka sen kaista menee pienemmän prioriteetin omaavan liikenteen hyötykäyttöön, jolloin verkon käyttöaste paranee. Ennen jokaista uutta mittausta, edellisessä mittauksessa tehdyt jonot on ensin poistettava.

```
root@raspberrypi-1:/home/pi# ovs-vsctl clear port eth0 qos
```

```
root@raspberrypi-1:/home/pi# ovs-vsctl --all destroy qos
root@raspberrypi-1:/home/pi# ovs-vsctl --all destroy queue
```

Tämän jälkeen luotiin mittausta 2 varten uudet jonot.

```
root@raspberrypi-1:/home/pi# ovs-vsctl set port eth0
qos=@newqos -- --id=@newqos create qos type=linux-htb
queues=1=@q1,2=@q2 -- --id=@q1 create queue other-config:min-
rate=2000000 other-config:priority=1 -- --id=@q2 create queue
other-config:min-rate=2000000 other-config:priority=2 -- --
id=@q3 create queue other-config:min-rate=2000000  other-con-
fig:priority=3
```

Kaikille kolmelle jonolle varattiin vähinään 2Mbit/s suuruinen kaista. Linux-htb jonotusalgoritmissa pienin prioriteetti arvo tarkoittaa korkeinta prioriteettia. Ruuhkatilanteen syntyessä, pakettien pitäisi purkautua ensimmäiseksi korkeimman prioriteetin jonosta eli tässä tapauksessa q1:stä.

9.2.4 Mittaus 3b: Yhden taatun kaistan jono

Koska mittauksessa 3b ei huomattu eroa edeltävään mittaukseen, haluttiin vielä mitata toimiko OVS:n vuorottelija paremmin silloin kun taatun kaistanleveyden omaavia jonoja on vain yksi, ja muut jonot toimivat leikkausperiaatteella.

```
ovs-vsctl set port eth0 qos=@newqos -- --id=@newqos create qos
type=linux-htb queues=1=@q1,2=@q2,3=@q3 -- --id=@q1 create queue
other-config:min-rate=2000000 -- --id=@q2 create queue other-con-
fig:max-rate=6000000 -- --id=@q3 create queue other-config:max-
rate=6000000
```

Jonolle 1 eli VOIP-liikenteelle varattiin 2Mbit/s kaista. Kaksi muuta jonoa toimivat leikkausperiaatteella mittauksen 2 mukaisesti.

9.2.5 Mittaus 4a: Purskeinen liikennevirta

Kaikissa aiemmissa mittauksissa kaikki liikennevirrat generoitiin tasaisina. Tämä ei vastaa aina todellisuutta, jossa tietoliikenne voi tulla purskeisena. Tämän takia mittauksessa 4 haluttiin testata, miten verkon toiminta muuttuu, jos yksi liikennevirroista (stream3) vaihdetaan purskeiseksi, ja palvelunlaatuasetukset pidetään samoina kuin mittauksessa 3. Liikennevirta3 pidettiin edelleen 5Mbit/s suuruisena ja sen purskeen kooksi valittiin 50kbit/s.

9.2.6 Mittaus 4b: Purskeprofiilin määrittely

Mittauksen 4a tulosten perusteella tiedettiin, miten purskeinen liikennevirta käyttäytyy verkossa. Mittauksessa 4b haluttiin testata miten purskeprofiilin luominen yhdellä jonoista vaikuttaa verkon toimivuuteen.

```
ovs-vsctl set port eth0 qos=@newqos -- --id=@newqos create qos
type=linux-htb queues=1=@q1,2=@q2,3=@q3 -- --id=@q1 create queue
other-config:min-rate=2000000 -- --id=@q2 create queue other-con-
fig:max-rate=6000000 -- --id=@q3 create queue other-config:max-
rate=6000000 other-config:burst=50000
```

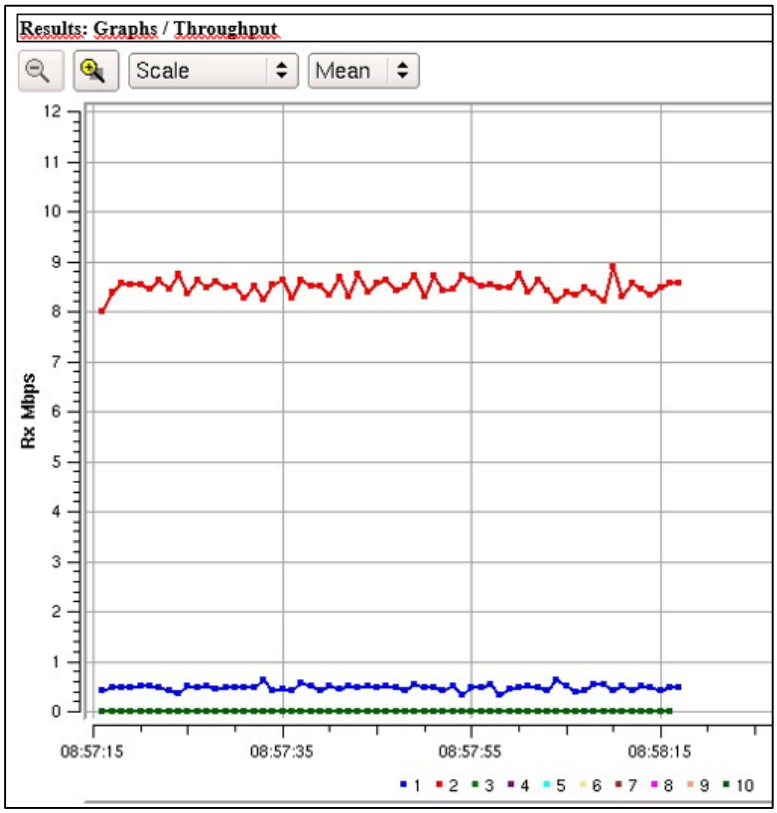
Jonot ovat muuten samat kuin edellisessä mittauksessa, mutta jonolle 3 luodaan 50kbit/s suuruinen purskeprofiili.

9.3 Tulokset

9.3.1 Mittaus 1

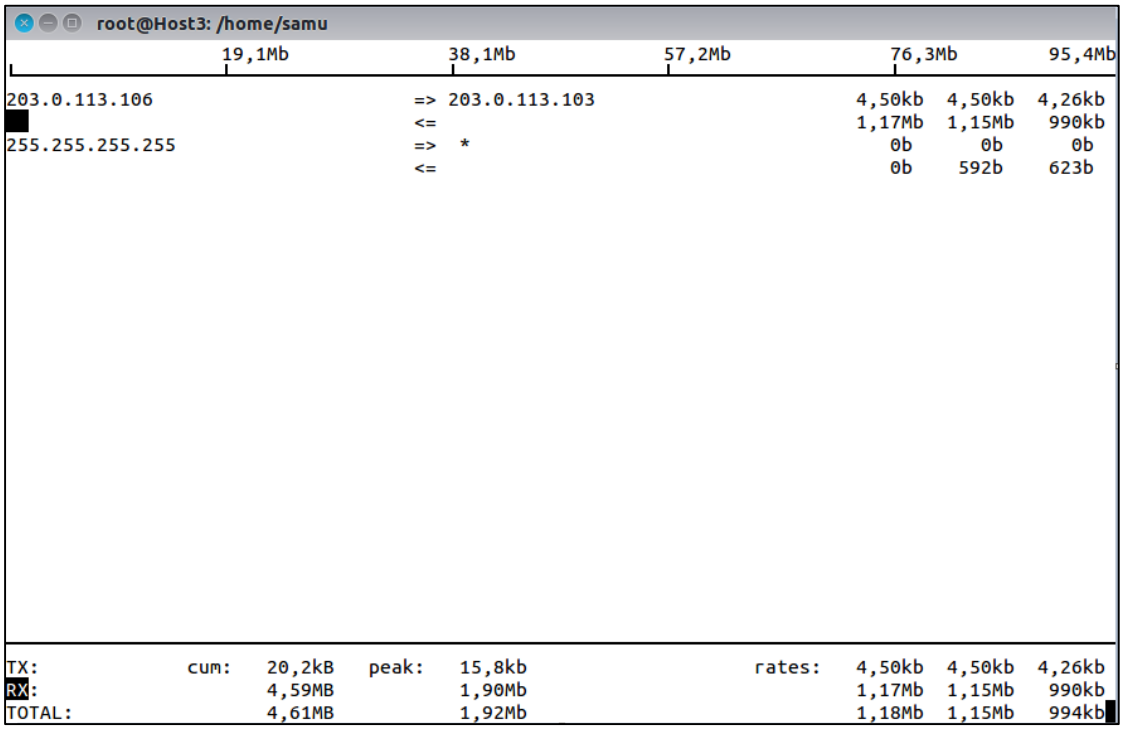
Mittauksen 1 tuloksista nähtiin, että ilman palvelun laatuasetuksia verkko toimi odotetulla tavalla. OVS käsitteli paketteja siinä järjestyksessä kuin niitä saapui ja käsitteli niitä tehtyjen dynaamisesti tehtyjen vuosääntöjen perusteella. Paketit ohjautuivat

samaan FIFO-periaatteella toimivaan jonoon. Tästä seurauksena liikennevirrat saivat kaistaa käyttöönsä suhteessa generoidun liikenteen määrään. Mittauksessa 1 huomattiin myös, että kontrolliyhteys Raspberrypi-1:n OVS-kytkimiltä pysyi yllä ruuhkatilanteesta huolimatta. Edes rajapinnan nopeuden pudottaminen 1Mbit/s ei riittänyt pudottamaan yhteyttä. Kuviossa 25 näkyy JDSUn tekemän raportin käyrä eri liikennevirtojen nopeuksista mittauksen aikana.



Kuvio 25. Mittaus 1 liikennevirtojen nopeudet

Käyrästä nähdään, että stream2 saa kaistaa yli 8Mbit/s kun stream 1 joutuu tyytymään alle yhteen. Stream3 mittausten tuloksia ei voida suoraan katsoa JDSUn raporteista, koska liikennevirran toisessa päässä ei ollut peilaavaa mittauslaitetta vaan tavallinen ubuntu-desktop virtuaalikone. Stream3 todennuksissa on käytetty Host3:lla ajetun iftop-ohjelman tuloksia, jotka mittauksen 1 osalta esitellään kuviossa 26.



Kuvio 26. Mittaus 1 Host3-iftop todennus

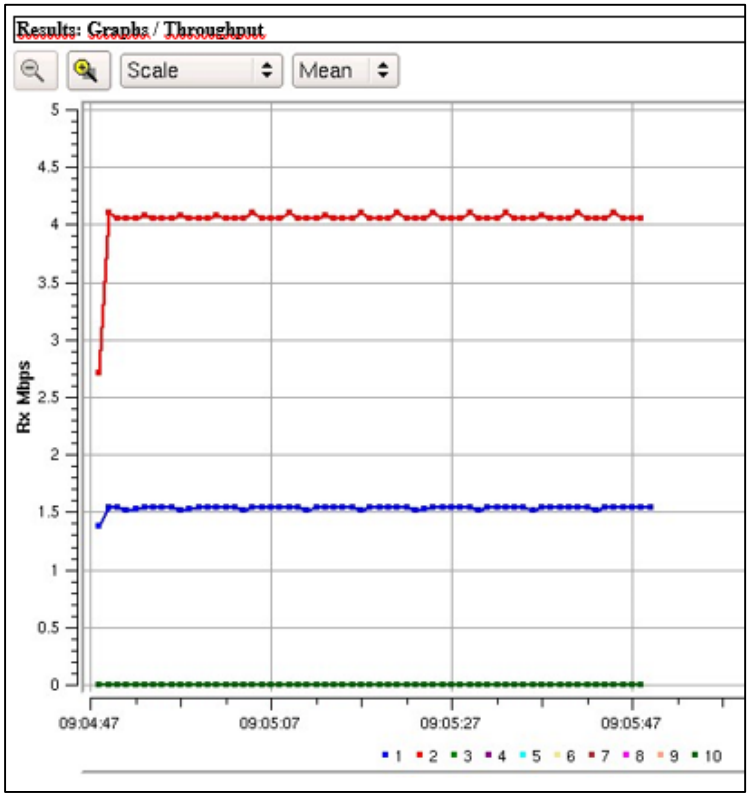
Stream3 sai kaistaa käyttöönsä hieman yli 1Mbit/s. Eri liikennevirtojen saamat kaistat ovat siis hyvin tarkasti samassa suhteessa niiden generoidun liikennemäärän kanssa. Mittauksen 1 tulosten yhteenveto esitetään taulukossa 5. Mittauksen 1 tarkemmat tulokset JDSUn raportista liikennevirtojen 1 ja 2 osalta löytyvät liitteestä 13.

Taulukko 5. Mittaus 1 tulokset

Liikennevirta	Lähetetty	Vastaanotettu
stream1	2mbit/s	0.47mbit/s
stream2	20mbit/s	8.46mbit/s
stream3	5mbit/s	1.15mbit/s

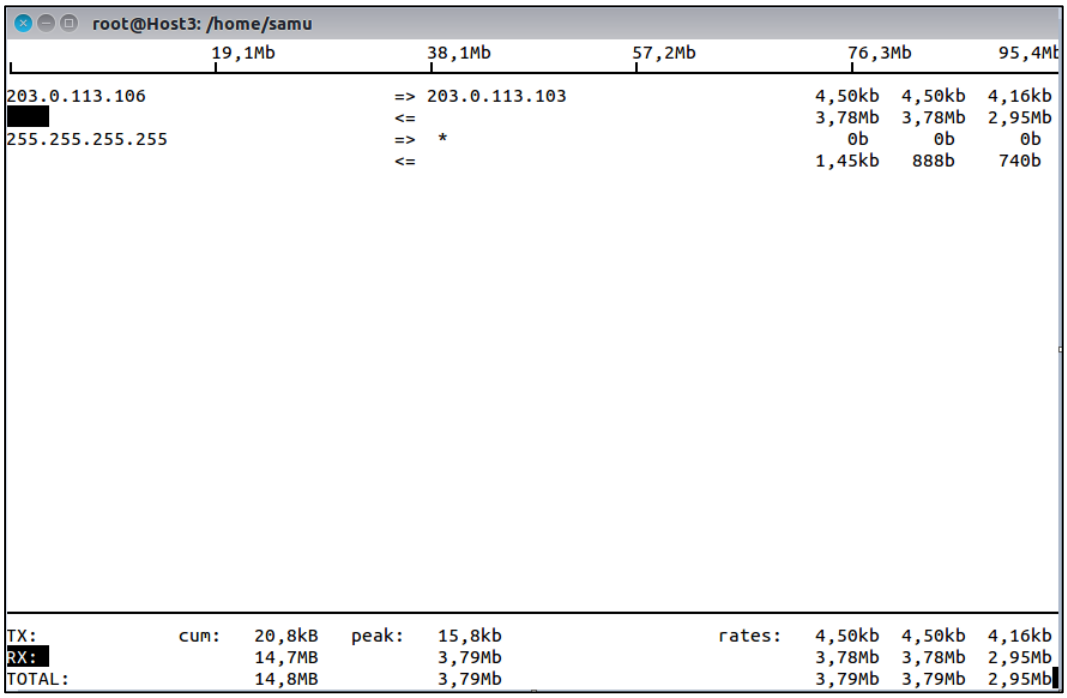
9.3.2 Mittaus 2

Mittauksen 2 tulokset olivat odotetun mukaiset. Liikenteen leikkaus onnistui, eikä yksikään kolmesta liikennevirrasta saanut enempää kaistaa, kuin mitä sille oli maksimissaan määritelty. Liikennevirtojen 1 ja 2 nopeudet ovat kuvattuna kuviossa 27.



Kuvio 27. Mittaus 2 liikennevirtojen nopeudet

Kuviossa 28 näkyy mittauksen 2 todennus Host3:lta katsottuna.



Kuvio 28. Mittaus 2 Host3-iftop todennus

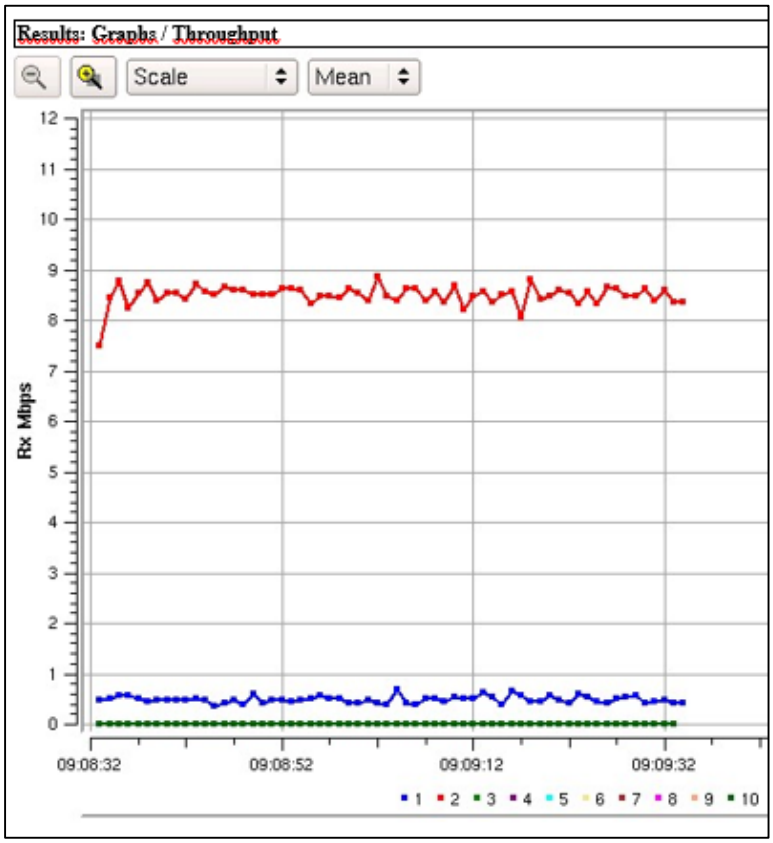
Todennuksista nähdään, että liikenteen leikkaus toimii halutulla tavalla. Mittauksen 2 tulosten yhteenveto esitetään taulukossa 6. Tarkemmat tulokset JDSUn raportista löytyvät liitteestä 14.

Taulukko 6. Mittaus 2 tulokset

Liikennevirta	Lähetetty	Vastaanotettu
stream1	2mbit/s	1.51mbit/s
stream2	20mbit/s	4.01/mbit/s
stream3	5mbit/s	3.78mbit/s

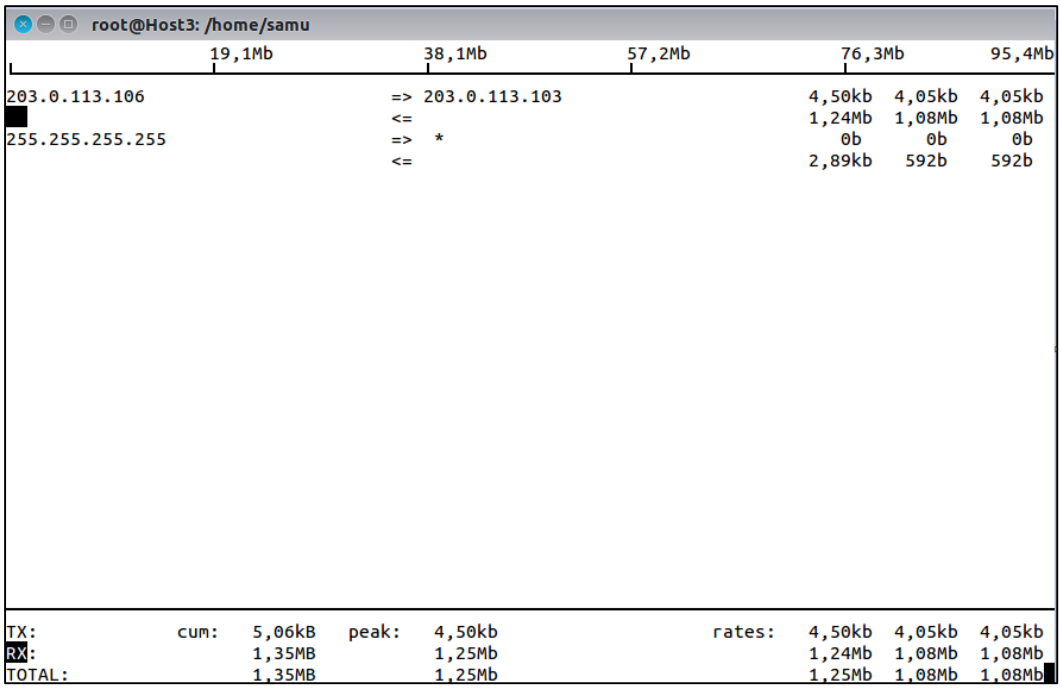
9.3.3 Mittaus 3a

Mittauksessa 3 huomattiin, että minimikaistan määrittely ei tuottanut haluttua tulosta. Mittauksen tulokset eivät eronneet mittauksen 1 tuloksista eli liikennevirtojen saama kaistanleveys määräytyi ainoastaan liikenteen määrän mukaan ja korkeamman prioriteetin liikenteelle ei pystytty takamaan kaistaa ruuhkatilanteessa. Kuviossa 29 näkyy liikennevirtojen nopeuksia osoittava käyrä mittauksen 3a osalta.



Kuvio 29. Mittaus 3a liikennevirtojen nopeudet

Kuviossa 30 näkyy vastaava todennus Host3:n iftop tulosteesta.



Kuvio 30. Mittaus 3a Host3-iftop todennus

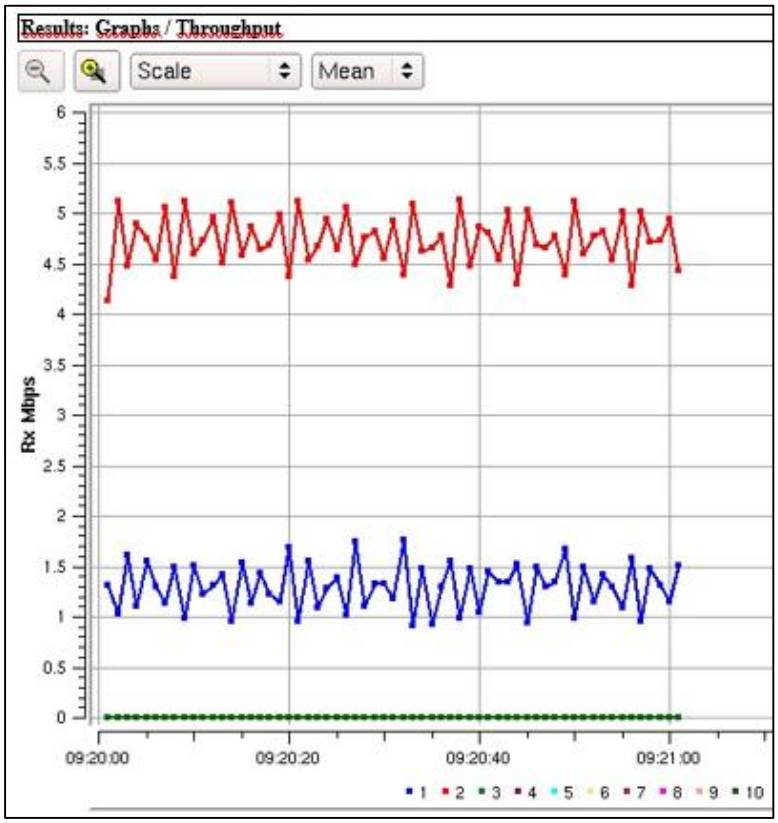
Todennuksista nähdään verkon käyttäytyvän identtisesti mittauksen 1 kanssa, joten mittauksen 3a perusteella voidaan sanoa, että minimikaistan rajaukset eivät toimineet. Mittauksen 3a tulosten yhteenveto esitetään taulukossa 7. Tarkemmat tulokset mittauksesta löytyvät liitteestä 15.

Taulukko 7. Mittaus 3a tulokset

Liikennevirta	Lähetetty	Vastaanotettu
stream1	2mbit/s	0.53mbit/s
stream2	20mbit/s	8.40mbit/s
stream3	5mbit/s	1.08mbit/s

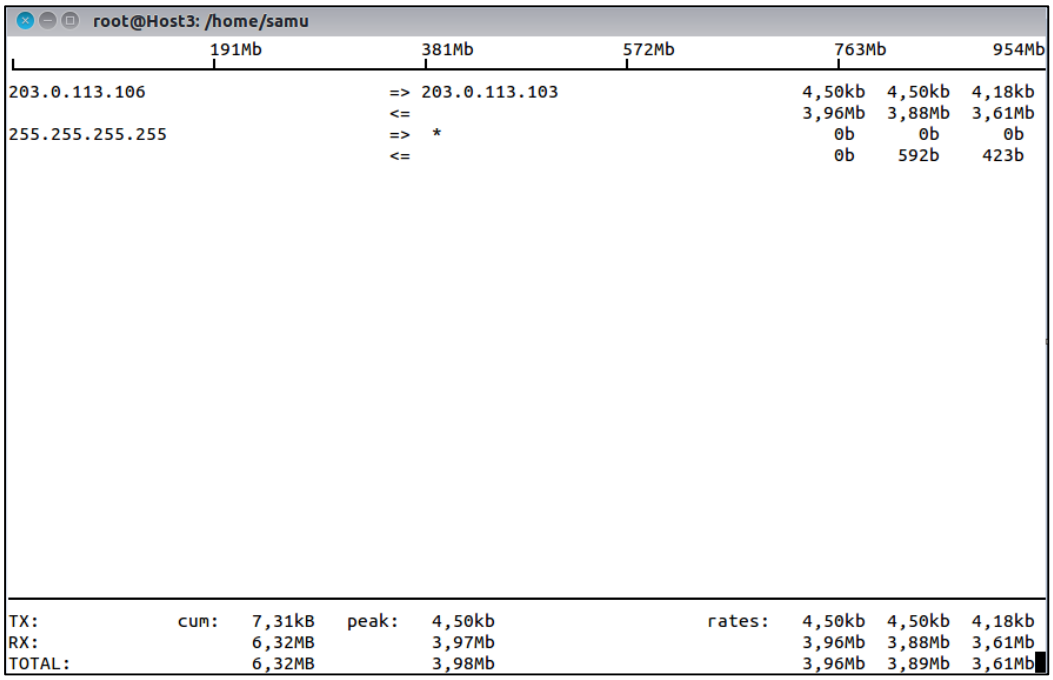
9.3.4 Mittaus 3b

Mittauksessa 3b huomattiin, että jos käytetään ainoastaan yhtä jonoa, jolle on määritelty minimikaista, jonojen vuorottelu toimii hieman paremmin, mutta ei kuitenkaan aivan halutulla tavalla. Kuviossa 31 näkyy liikennevirtojen nopeuskäyrät mittauksen 3b osalta.



Kuvio 31. Mittaus 3b liikennevirtojen nopeudet

Kuviossa 32 näkyy vastaava todennus Host3:lta katsottuna.



Kuvio 32. Mittaus 3b Host3-iftop todennus

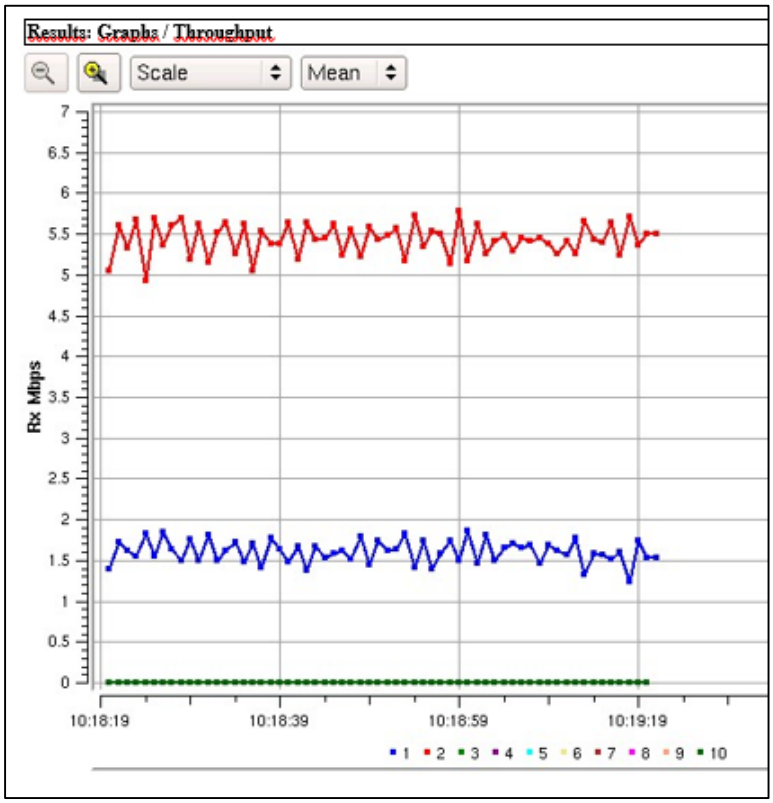
Todennuksista voidaan havaita, että mittaukseen 3a verrattuna liikennevirta2 saa 1-1,5Mbit/s kaistaa, 0,8Mbit/s verrattuna. Kaistan varaus toimii siis jossain määrin, mutta haluttuun 2Mbit/s ei vieläkään päästä. On kuitenkin huomattavaa, että liikennevirran 1 lisäksi myös liikennevirran 3 saama kaista kasvoi huomattavasti. Mittauksen 3b tulokset esitetään taulukossa 8. Tarkemmat tulokset JDSU-raportista löytyvät liitteestä 16.

Taulukko 8. Mittaus 3b tulokset

Liikennevirta	Lähetetty	Vastaanotettu
stream1	2mbit/s	1.43mbit/s
stream2	20mbit/s	4.42mbit/s
stream3	5mbit/s	3.88mbit/s

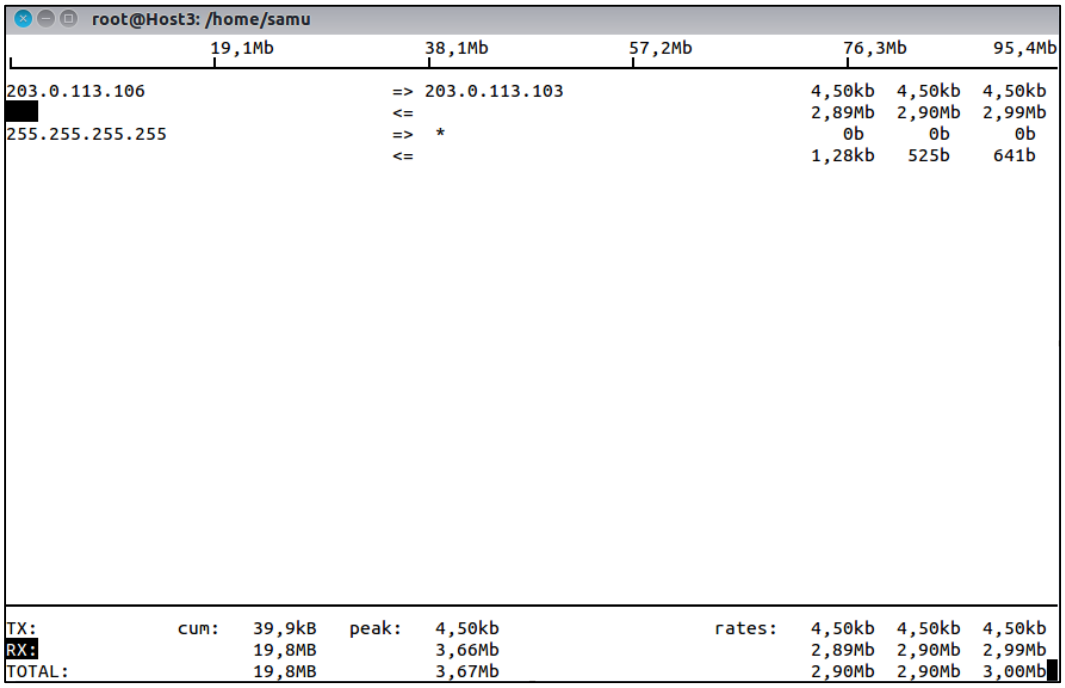
9.3.5 Mittaus 4a

Mittauksen 4 tulokset ovat sikäli hieman epätarkat, että purskeinen liikennevirta lähetettiin Host3:lle eikä JDSUn vastaanottaville laitteille. Tämän seurauksena tarkkoja arvoja ei liikennevirran 3 osalta saatu vaan tulokset on pääteltävä muiden liikennevirtojen tuloksista. Kuviossa 33 näkyy liikennevirtojen nopeudet mittauksen 4a osalta



Kuvio 33. Mittaus 4a liikennevirtojen nopeudet

Kun mittauksen tuloksia verrataan mittauksen 3b tuloksiin, huomataan että liikennevirrat 1 ja 2 ovat saaneet keskimäärin noin 15 % suuremman tiedonsiirtonopeuden. Tästä voidaan päätellä, että purskeinen liikennevirta kärsii ruuhkatilanteissa, mikäli purskeprofiilia ei ole määritelty. Tämä nähdään myös katsottaessa Host3:n iftop-tulostetta, joka löytyy kuviosta 34.



Kuvio 34. Mittaus 4a Host3-iftop todennus

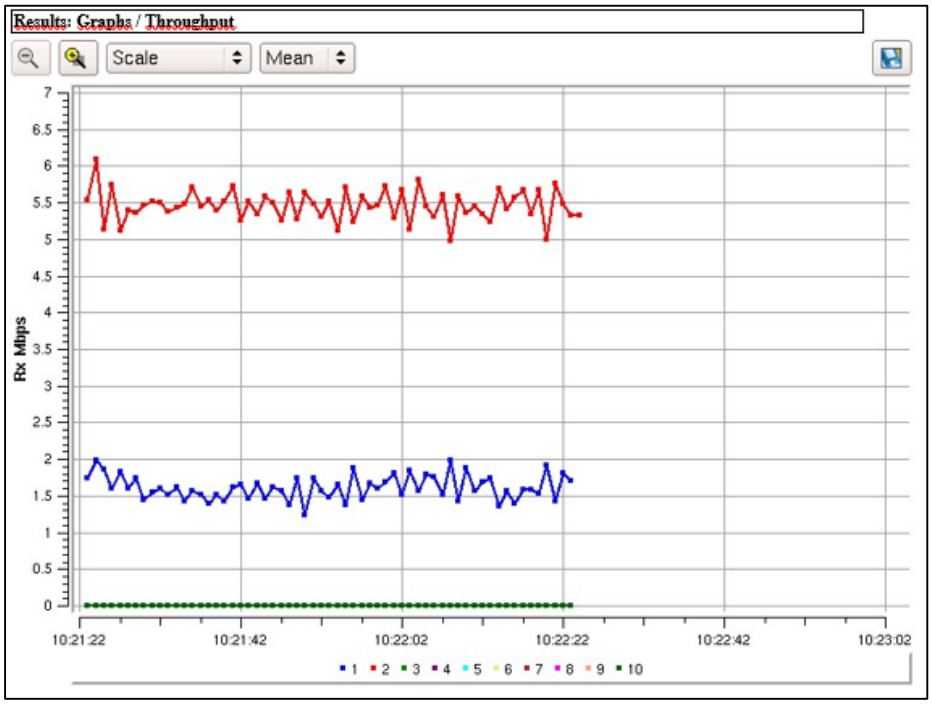
Kuviosta voidaan myös todeta Host3:n saaman kaistanleveyden pudonneen noin 1mbit/s. Mittauksen 4a tulokset esitetään taulukossa 9. Tarkemmat tulokset JDSUn raportista löytyvät liitteestä 17.

Taulukko 9. Mittaus 4a tulokset

Liikennevirta	Lähetetty	Vastaanotettu
stream1	2mbit/s	1.55mbit/s
stream2	20mbit/s	5.39mbit/s
stream3	5mbit/s	2.89mbit/s

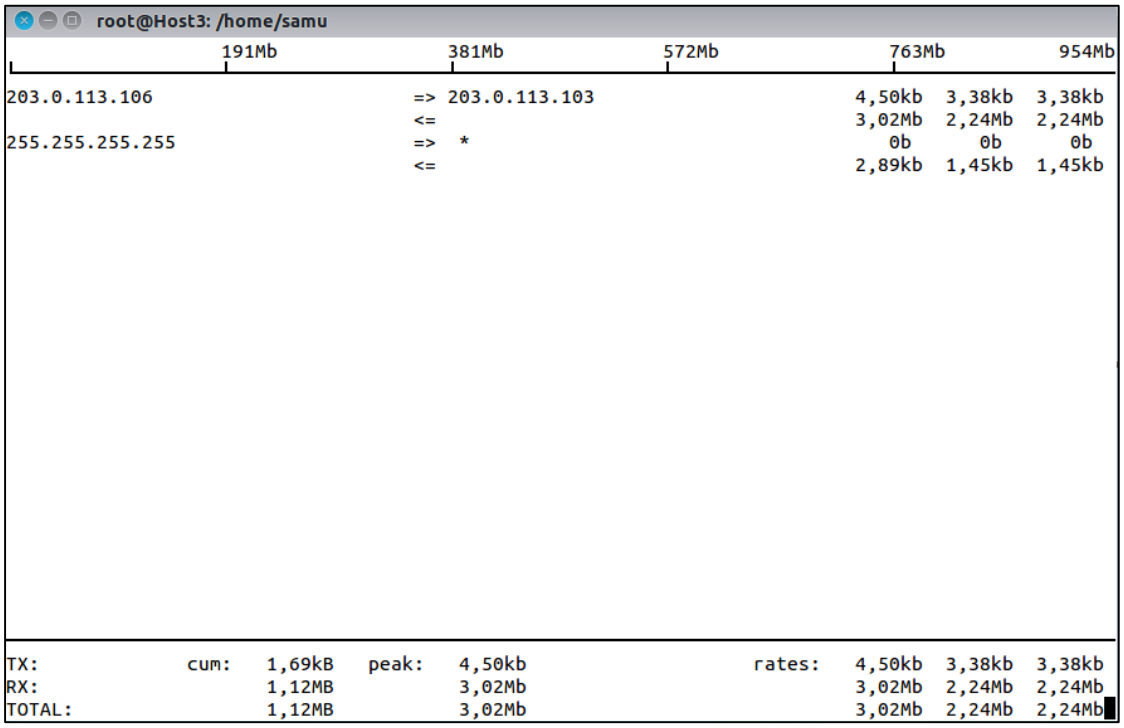
9.3.6 Mittaus 4b

Tuloksista voidaan huomata, että liikennevirtojen keskimääräisiin nopeuksiin ei tullut suurta muutosta purskeprofiilin määrittelyllä, mutta nopeuksissa tapahtuneet vaihtelut ovat olleet suurempia. Liikennevirtojen nopeudet mittauksessa 4b näkyvät kuvissa 35.



Kuvio 35. Mittaus 4b liikennevirtojen nopeudet.

Kun liitteessä 18 olevia tuloksia katsotaan tarkemmin, huomataan että viivenvaihtelu on selvästi kasvanut liikennevirtojen 1 ja 2 kohdalla. Eli kun purskeista liikennettä on saapunut enemmän, purskeprofiilin ylimääräinen kaista on tullut liikennevirran 3 käyttöön, ja muiden liikennevirtojen viiveet ovat näillä hetkillä kasvaneet. Pidemmällä ajanjaksolla keskimääräisessä kaistanleveydessä liikennevirtojen välillä ei ole suuria eroja kuten myös kuviosta 36 huomataan.



Kuvio 36. Mittaus 4b Host3-iftop todennus

Mittauksen 4b tulosten yhteenveto esitetään taulukossa 10. JDSUn raportti löytyy liitteestä 18.

Taulukko 10. Mittaus 4b tulokset

Liikennevirta	Lähetetty	Vastaanotettu
stream1	2mbit/s	1.65mbit/s
stream2	20mbit/s	5.28mbit/s
stream3	5mbit/s	3.02mbit/s

9.4 Mittaustulosten pohdinta

Mittausten tuloksena on, että RYUn tekemät DSCP-merkkaus ja jonoihin ohjaavat vuosäännöt toimivat ja jonojen luominen OVS:llä on mahdollista. Jonoihin määritelty leikkaus toimi odotusten mukaisesti tietyn tiedonsiirtonopeuden ylittyessä. Minimi-

kaistan rajaus OVS:llä ei onnistunut toivotusti ja ainoastaan yhdellä jonolla, jolle minimikaista oli määritelty, saatiin edes jonkinlaista palvelun laatua edistävää muutosta havaittua. PurskeprOFiilin mittauksen suhteen tehtiin virhe, kun purskeista liikennettä ei lähetetty JDSU:n mittauslaitteelle, jolloin olisi saatu raportuitua tarkempaa dataa, mutta tämän työnkin mittauksen perusteella voidaan sanoa myös purskeprofiilin toimivan ainakin jossain määrin. OVS:n TCP-yhteyksiä kontrollereihin ei pystytty priorisoimaan, mutta tämä ei työssä tehdyissä mittausskenaarioissa muodostunut ongelmaksi. Mikäli kontrolleriyhteyksiä olisi ollut useampia ja olisivat itsessään vieneet suuremman osan kaistasta ja muodostettu ruuhkatilanne olisi ollut tämän työn mittauksia vakavampi, olisi ongelmia voinut muodostua.

Työn teoriaosuudessa käsiteltiin mm. OFin tarjoamia palvelun laatu mahdollisuuksia. OFin kahdesta palvelun laatuun sunnitellusta työkalusta, OVS tukee ainoastaan jonojen konfigurointia. OVS:llä on kyllä OF-tuki meterienkin konfigurointiin, mutta sitä ei ole implementoitu OVS-kytkimeen. Avoimen lähdekoodin SDN-kontrollereissa havaittiin puutteita tiettyjen vuosääntöjen konfiguroinnissa, joka kävi ilmi Opendaylightin ja Floodlightin set_queue actionin lisäämisessä OVS:n vuotauluun. Opendaylight lisäsi kyllä vuosäännön omaan tietokantaansa, mutta OVS:n vuotauluun se ei toimintoa lisännyt, vaikka vuosääntö muuten menikin perille. ONOSista taas puuttuu mahdollisuus muokata IP-paketin DSCP-kenttää pakettien merkkausta varten. RYUlla onnistui sekä pakettien merkkaus, että jonoihin ohjaus, ilman ongelmia. RYUllakin on kuitenkin ongelma, että näitä toimintoja tekevä rest_qos-applikaatio on täysin erillinen muista applikaatioista, joten sen sovittaminen verkkoon, jossa haluttaisiin käyttää useita erilaisia RYUn applikaatioita (firewall, snort, router jne.), voi tuottaa ongelmia vuotaulujen monimutkaistuessa ja aiheuttaessa mahdollisia ristiriitoja toisten applikaatioiden tekemien vuosääntöjen kanssa. Esimerkiksi rest_qos-applikaatio käyttää OVS:n tunnistamiseen käytettyä datapath_id:tä hexadesimaalimuodossa, kun taas osa RYUn applikaatioista käyttää sitä desimaalimuodossa. Tämä tuo turhaa monimutkaisuutta laitteiden konfigurointiin. RYUsta puuttuu myös käyttäjäystävällinen käyttöliittymä ja vuosäännöt on asetettava kytkimelle käyttämällä curlia. Esimerkiksi rest_qos-applikaation tekemien vuotietojen hakeminen RYUlta käsin ei onnistunut, vaan vuotaulut piti katsoa OVS:ltä käsin.

Vaikka jonojen luominen OVS:llä on mahdollista, OVS:n jonotusalgoritmin implementoinnissa on kuitenkin puutteitta silloin kun tietylle jonolle halutaan kaikissa tilanteissa varata tietty kaistanleveys. Jos OVS:lle tulee tulevaisuudessa meter-implementaatio, tämä ongelma saattaa korjautua, koska meterin pitäisi mitata koko ajan sisääntulevaa liikennettä ja reagoida myös määrättyjen rajojen alitukseen. Yhteenvetona voidaan todeta, että SDN-verkkotekniikka teoriassa mahdollistaa verkon päästä päähän toteutuvan palvelunlaadun toteutumisen dynaamisemmin ja skaalautuvammin kuin perinteiset tietoverkot. Myös OF-protokollan kehittelyssä palvelun laatu vaatimukset ovat selvästi otettu huomioon. Avoimen lähdekoodin OF-kontrollerit ja kytkimet ovat kuitenkin kaikki sen verran keskeneräisiä, että palvelun laadun hienojakoisuudessa ja skaalautuvuudessa on vielä puutteita verrattuna perinteisiin verkkolaitteisiin.

10 SDN-testbedin fyysinen laajennus

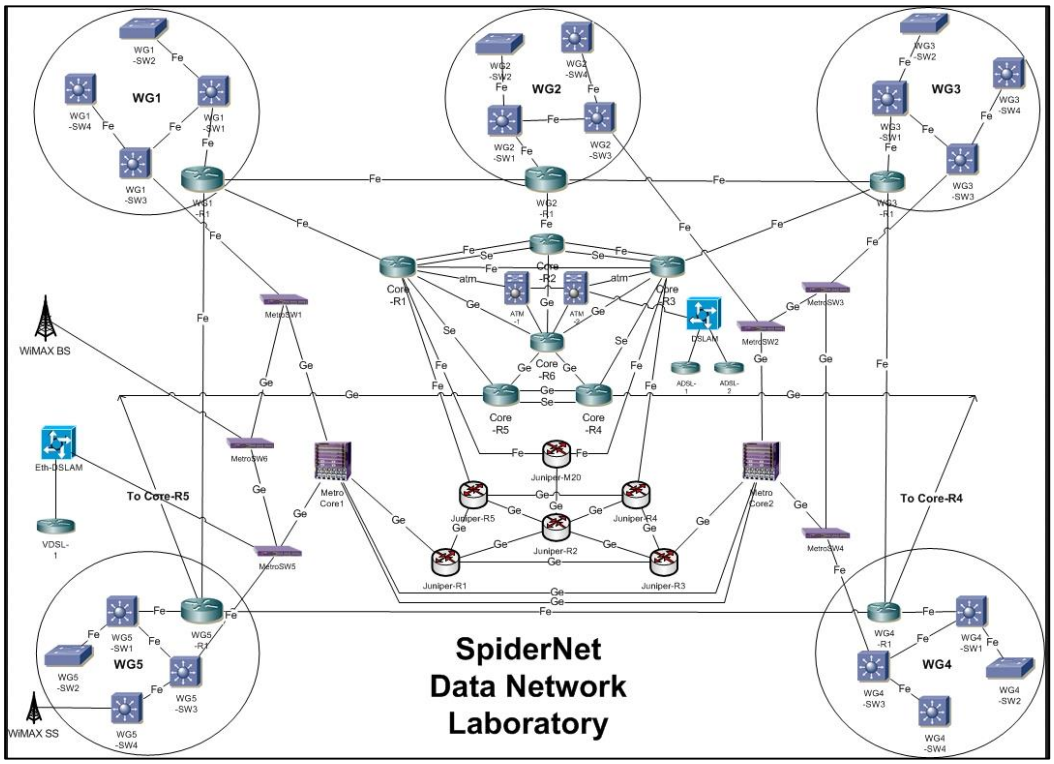
10.1 Tavoitteet ja suunnittelu

SDN-verkkojen palvelun laadun tutkimisen jälkeen, haluttiin laajentaa Cyber Trust – projektin virtualisoitua SDN-testbed ympäristöä fyysisillä laitteilla. Ympäristön laajennuksen on tarkoitus toimia pohjana ympäristölle, jota JAMK voi tulevaisuudessa käyttää opetuskäytössä esim. palvelun laatu opintojakson laboratorioharjoituksissa. Verkon toteutukseen hyödynnetään sekä JYVSECTECin RGCE-ympäristöä, että JAMKin SpiderNet-laboratorioverkkoa.

SDN-testbed virtuaaliympäristö on toteutettu JYVSECTECin RGCE-kybertoimintaympäristöön. RGCE on internetiä mallintava, suljettu tutkimus- kehitys – ja koulutusympäristö. RGCE on rakennettu vastaamaan mahdollisimman tarkasti oikeaa internetiä, jotta sillä voidaan tehdä realistista tutkimustyötä. Se koostuu useiden verkko-operaattoreiden verkoista, jotka tarjoavat kaikki tarvittavat internetin ydinpalvelut ja mahdollisuuden generoida realistista verkkoliikennettä järjestelmien

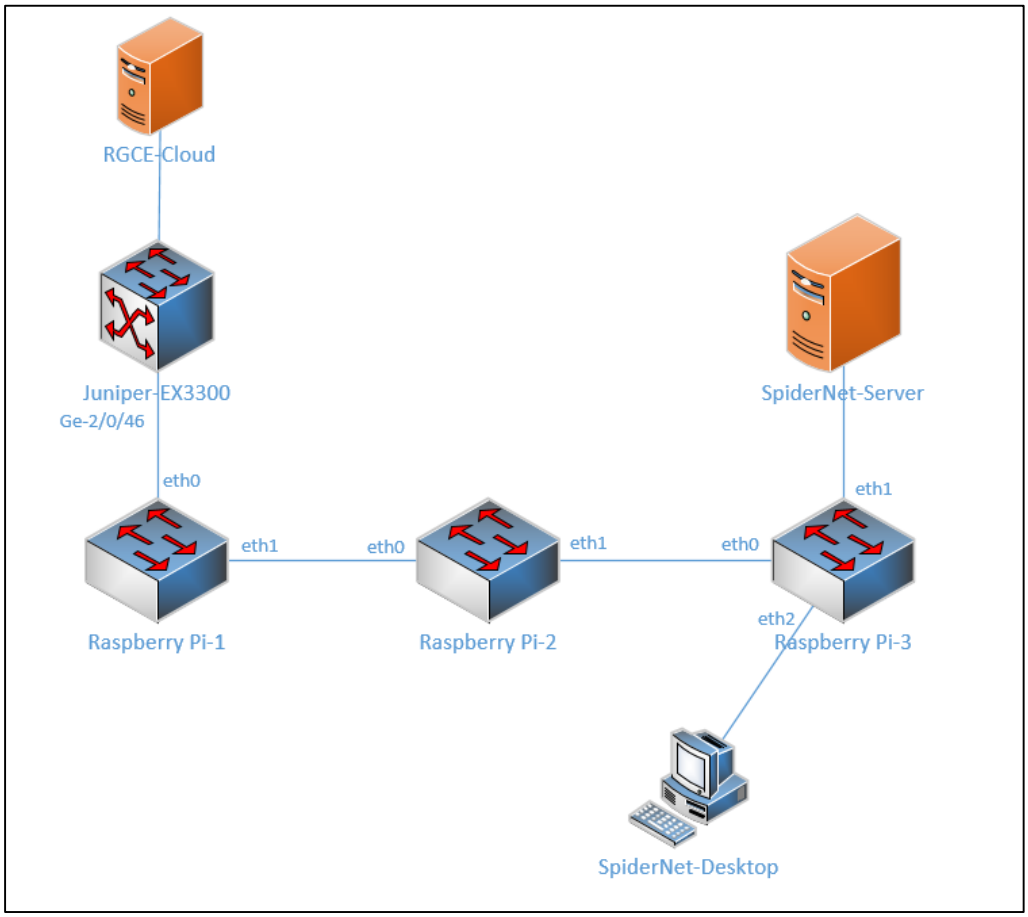
testaukseen. Suljettu ympäristö mahdollistaa erilaisten järjestelmien turvallisen testauksen ilman tietoturvariskiä (RGCE-Kybertoimintaympäristö n.d.)

SpiderNet on JAMKin tuotantoverkosta eristetty laboratorioverkko, jota käytetään verkko-operaattoreiden ja palveluntarjoajien teknologioiden testaus ja opetuskäyttöön. Se koostuu useiden eri laitevalmistajien laitteista (Cisco Systems, Juniper Networks, Extreme Networks, Airspan Networks ja Zhone) mahdollistaakseen mahdollisimman monen eri protokollan ja teknologian tutkimisen (SpiderNet 2009.) SpiderNetin fyysinen topologia kokonaisuudessaan esitetään kuviossa 37.



Kuvio 37. SpiderNet fyysinen topologia (Spidernet 2009)

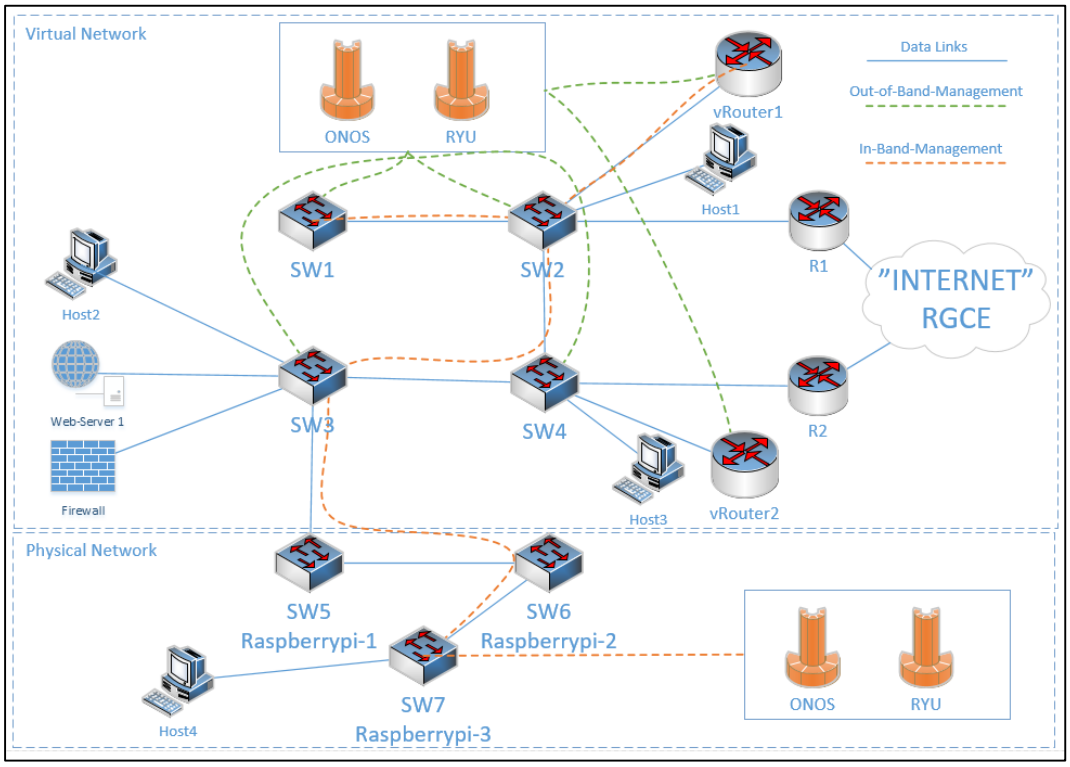
SDN-testbedin fyysinen laajennus toteutettiin kolmella Raspberry Pillä. Ne yhdistettiin RGCE-ympäristöön SpiderNetin laitteiden kautta. Verkon laajennuksen fyysinen topologia esitetään kuviossa 38.



Kuvio 38. SDN-testbed fyysinen laajennus

Raspberrypi-1 yhdistetään SpiderNetissä sijaitsevaan Juniper-EX3300-sarjan kytkimeen. Tämän kytkimen lävitse tarvitaan liitäntää RGCE-ympäristöön SW3-to-Raspi1-verkkoon. Liikenne kulkee Raspberrypi-1:n ja EX3300:n välillä leimaamattomana, jonka jälkeen EX3300 lisää kehyksiin VLAN-leiman liikenteen erottamiseksi.

SDN-testbedin looginen topologia laajennuksen jälkeen esitetään kuviossa 39.



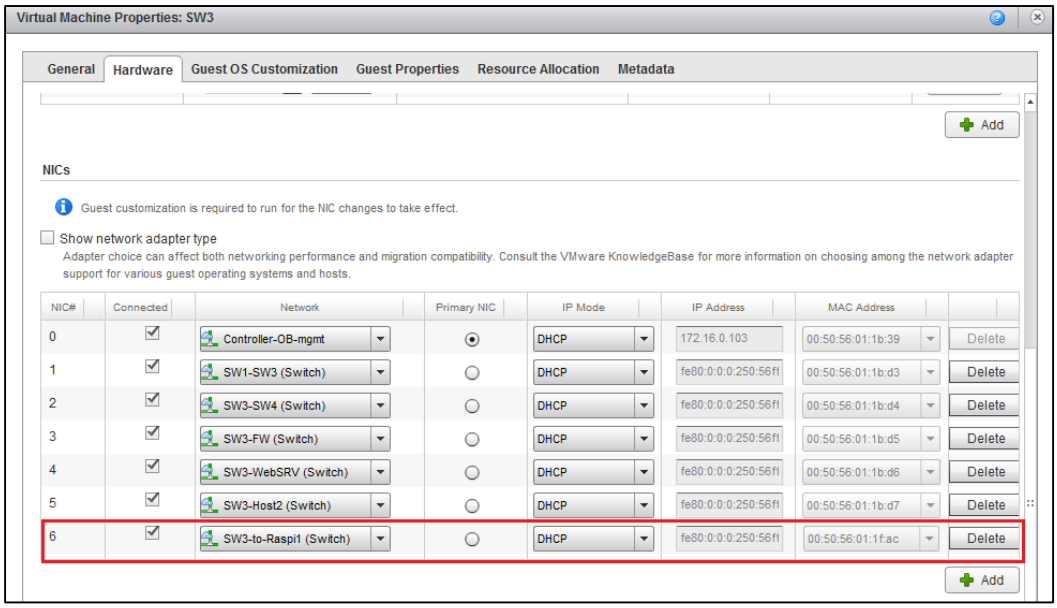
Kuvio 39. SDN-testbed looginen topologia

Ennen tämän työn aloittamista, SDN-testbedin virtuaaliympäristö koostui useasta NFV-komponentista (kytkimet, reitittimet, palomuuri), joita pyöritetään erillisillä virtuaalikoneilla. Tämän lisäksi ympäristössä on kontrolleriverkko, jossa on eri SDN-kontrollereita, joilla voidaan ajaa erilaisia testiskenaarioita. Laajennuksella lisättiin OVS-kytkinten ja päätelaitteiden määrää. Kuten ylläolevasta kuviosta nähdään, Raspberry Piiden kontrolliliikenne menee In-Band-Management-liikenteenä reitittimen kautta, kun taas RGCEn kytkimet ottavat yhteyden kontrolleriverkkoon erillistä hallintaverkkoa pitkin. Koska RGCE on täysin eristetty ympäristö, ei sen julkisilla IP-osoiteavaruuksilla tai AS-alueilla ole mitään yhteyksiä oikean internetin vastaavien osoitteiden palveluihin tai omistajiin. SDN-testbedissä on käytetty julkista IP-osoiteavaruutta 203.0.113.0/24. Kaikkien SDN-testbedissä käytettyjen laitteiden rajapintojen IP – ja MAC-osoitteet löytyvät liitteestä 19.

10.2 Toteutus

10.2.1 Fyysisten laitteiden liittäminen RGCE-ympäristöön

SDN-testbedin fyysistä laajennusta varten SW3:lle lisättiin uusi virtuaaliverkko kuvissa 40 näkyvällä tavalla.



Kuvio 40. SW3-to-Raspi1 verkon luominen

Tämän verkon kautta liikenne ohjataan SpiderNetiin. Tämän jälkeen jokaiselle Raspberry Pille luotiin OVS-kytkin ja siihen lisättiin tarvittavat rajapinnat ja kytkimelle annettiin IP-osoite. OVS-prosessi haluttiin myös laittaa käynnistymään bootin yhteydessä.

```
root@raspberrypi-1:/home/pi# ovs-vsctl add-br OF-switch
root@raspberrypi-1:/home/pi# ovs-vsctl add-port OF-switch eth0
root@raspberrypi-1:/home/pi# ovs-vsctl add-port OF-switch eth1
root@raspberrypi-1:/home/pi# ovs-vsctl add-port OF-switch eth2
root@raspberrypi-1:/home/pi# ifconfig OF-switch 172.16.1.101/24
root@raspberrypi-1:/home/pi# update-rc.d openvswitch-switch de-
faults
```

Bootin yhteydessä OF-switchille pitää antaa myös IP-osoite ja luoda staattinen reitti hallintaverkkoon 172.16.0.0/24 vRouter1:n kautta. Tämä toteutettiin ajamalla komennot ajastettuna crontabilla. Komennot lisättiin tiedostoon /etc/crontab

```
@reboot root sleep 30 && ifconfig OF-switch 172.16.1.101/24 &&
ip route add default via 172.16.1.1
```

Vastaavat konfiguraatiot tehtiin myös kahdelle muulle Raspberry Pille. Niiden konfiguraatiot löytyvät liitteistä 20 ja 21.

Tämän jälkeen OF-switch yhdistettiin RGCEssä sijaitsevaan kontrolleriin. Ensin Raspberry Piit yritettiin yhdistää ONOSiin, koska sitä on käytetty eniten RGCE-ympäristön OVS-kytkinten hallintaan.

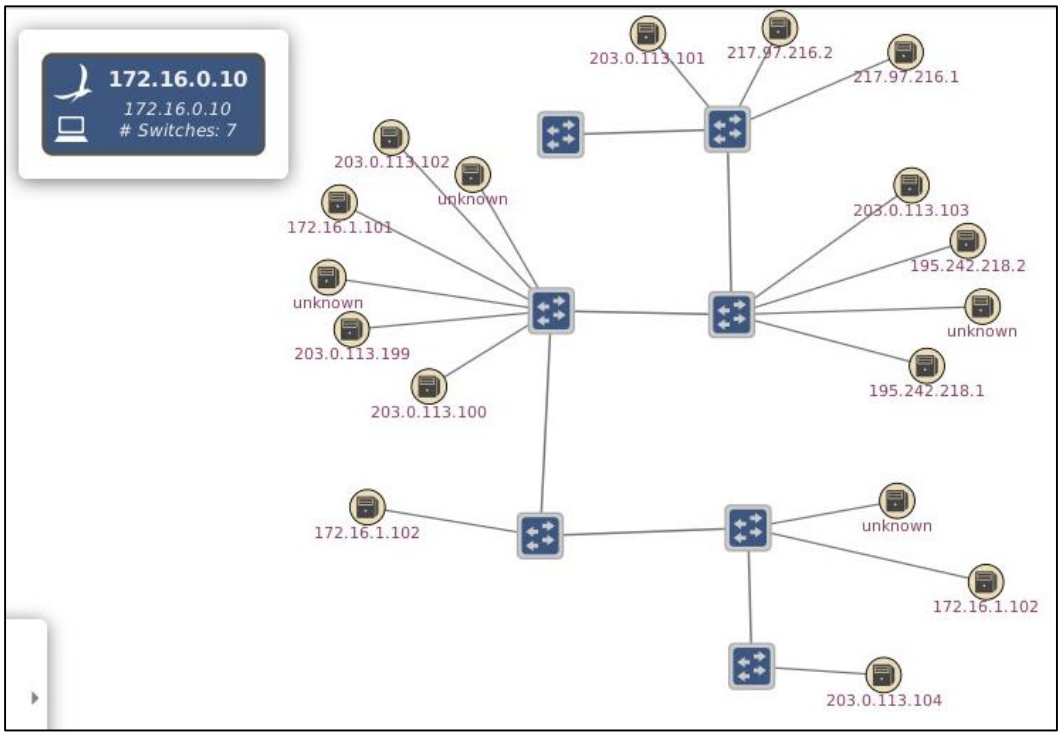
```
root@raspberrypi-1:/home/pi# ovs-vsctl set-controller OF-switch
tcp:172.16.0.10:6633
```

OVS:t SW5-SW7 yrittävät tässä vaiheessa lähettää ARP-request-viestin vRouter1:lle. Tämä ARP-viesti ei kuitenkaan mene perille, koska RGCEn kytkimet tiputtavat kehyksen, jos niiden ARP-taulusta ei löydy molempia laitteita. Asia korjaantuu lähettämällä ICMP-viestejä vRouter1:ltä SW5:lle. Kuviossa 41 näkyy, että liikenne alkaa kulkea laitteiden välillä muutaman epäonnistuneen paketin jälkeen.

```
vyos@vRouter1# ping 172.16.1.101
PING 172.16.1.101 (172.16.1.101) 56(84) bytes of data.
64 bytes from 172.16.1.101: icmp_req=4 ttl=64 time=2008 ms
64 bytes from 172.16.1.101: icmp_req=5 ttl=64 time=1008 ms
64 bytes from 172.16.1.101: icmp_req=6 ttl=64 time=8.65 ms
64 bytes from 172.16.1.101: icmp_req=7 ttl=64 time=1.43 ms
64 bytes from 172.16.1.101: icmp_req=8 ttl=64 time=1.14 ms
64 bytes from 172.16.1.101: icmp_req=9 ttl=64 time=0.927 ms
64 bytes from 172.16.1.101: icmp_req=10 ttl=64 time=1.07 ms
64 bytes from 172.16.1.101: icmp_req=11 ttl=64 time=1.15 ms
^C
--- 172.16.1.101 ping statistics ---
45 packets transmitted, 8 received, 82% packet loss, time 44016ms
rtt min/avg/max/mdev = 0.927/378.910/2008.205/698.376 ms, pipe 3
[edit]
vyos@vRouter1# _
```

Kuvio 41. vRouter1 ping SW5 todennus

Kun ICMP-viestit menevät läpi, RGCEn kytkin päästä ARP-viestit läpi ja SW5 saa muodostettua yhteyden ONOSiin. Jostain syystä nämä kytkimet lakkaavat vastaamasta ICMP-viesteihin kontrolleriyhteyden muodostettua, mutta tämä ei kuitenkaan vaikuta liikenteen ohjaukseen. Kun vRouter1:ltä on pingattu myös SW6 ja SW7 Verkkotopologiaa voidaan tarkastella myös ONOSin graafisen käyttöliittymän kautta osoitteesta http://172.16.0.10:8181/onos/ui. ONOSin graafinen käyttöliittymä näkyy kuviossa 42



Kuvio 42. ONOS graafinen käyttöliittymä.

Kun katsotaan ONOSin graafista käyttöliittymää, voidaan sen todeta vastaavaan kuviossa 39 esitettyä verkon loogista topologiaa. ONOSin graafinen käyttöliittymä piirtää kaikki OF-kytkimet, jotka ovat siihen yhteydessä ja niihin liitetyt päätelaitteet. Kuviossa näkyy myös, että mikäli OVS-kytkimelle laitetaan IP-osoite, kuten tässä tapauksessa SW5-SW7 kytkimille on tehty, se piirtää nämä IP-osoitteet päätelaitteina. Myös

Spidernetissä sijaitseva Host4 näkyy kuvassa, mutta kuten kuviosta 43 voidaan kuitenkin todeta, Host4 ei kuitenkaan saa yhteyttä RGCEssä sijaitseviin, samaan aliverkkoon kuuluviin päätelaitteisiin.

```
administrator@SP-Ubuntu:~$ ping 203.0.113.101
PING 203.0.113.101 (203.0.113.101) 56(84) bytes of data.
From 203.0.113.104 icmp_seq=1 Destination Host Unreachable
From 203.0.113.104 icmp_seq=2 Destination Host Unreachable
From 203.0.113.104 icmp_seq=3 Destination Host Unreachable
From 203.0.113.104 icmp_seq=4 Destination Host Unreachable
From 203.0.113.104 icmp_seq=5 Destination Host Unreachable
From 203.0.113.104 icmp_seq=6 Destination Host Unreachable
From 203.0.113.104 icmp_seq=7 Destination Host Unreachable
From 203.0.113.104 icmp_seq=8 Destination Host Unreachable
From 203.0.113.104 icmp_seq=9 Destination Host Unreachable
^C
--- 203.0.113.101 ping statistics ---
10 packets transmitted, 0 received, +9 errors, 100% packet loss, time 9047ms
pipe 3
administrator@SP-Ubuntu:~$
```

Kuvio 43. Epäonnistunut ping Host1->Host4 todennus

Host4:n lähettämä ARP-request saapuu SW3-kytkimelle, mutta sen kysyessä ONOSilta, mitä tälle kehykselle kuuluu tehdä, ONOS ei lähetä vastausta. Kuviossa 44 näkyy SW-3 kytkimen lähettämä OF-viesti.

```
No.   Time          Source          Destination     Protocol  Length  Info
  926 5.141693000   172.16.0.103    172.16.0.10     OpenFlow     168  Type: OFPT

  ▸ Ethernet II, Src: Giga-Byt_11:01:66 (00:16:e6:11:01:66), Dst: Broadcast (ff:ff:ff:ff:
  ▾ Address Resolution Protocol (request)
      Hardware type: Ethernet (1)
      Protocol type: IP (0x0800)
      Hardware size: 6
      Protocol size: 4
      Opcode: request (1)
      Sender MAC address: Giga-Byt_11:01:66 (00:16:e6:11:01:66)
      Sender IP address: 203.0.113.104 (203.0.113.104)
      Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
      Target IP address: 203.0.113.101 (203.0.113.101)
```

Kuvio 44. ONOS OFPT_Packet_in Wireshark-kaappaus.

Kuviossa näkyy OFPT_Packet_in OF kehys, jossa SW3 kysyy ONOSilta, mitä Host4:lta saapuvalle ARP Request-viestille tehdään. ONOS pitäisi vastata tähän viestiin OFPT_Packet-out-viestillä, jossa se kertoo, että ARP-request lähetetään ulos kaikista

porteista, paitsi sinne mistä se on tullut. ONOS ei kuitenkaan selviämättömästä syystä koskaan lähettänyt tähän viestiin vastausta, jolloin SW3 pudotti Host4:lta saapuvat paketit. Tämä ongelma ilmeni nimenomaan ONOSin yhteydessä silloin kun siirryttiin RGCE-ympäristöstä SpiderNetiin tai toisinpäin. Eli RGCEstä Host4:lle menevä liikenne tippui samalla tavalla SW5:n kohdalla, kun ONOS ei vastannut sille, mitä ARP Request-viestille olisi pitänyt tehdä. Koska ongelma näytti olevan kontrollerikohtainen, päätettiin kokeilla verkon toimivuutta RYU-kontrollerin kanssa. RYUlta käynnistettiin ainoastaan simple_switch_13-applikaatio liikenteen ohjaukseksi.

```
root@ryu~# ryu-manager ryu.app.simple_switch_13
```

Tämän jälkeen RYU asetettiin jokaisen OVS-kytkimen kontrolleriksi.

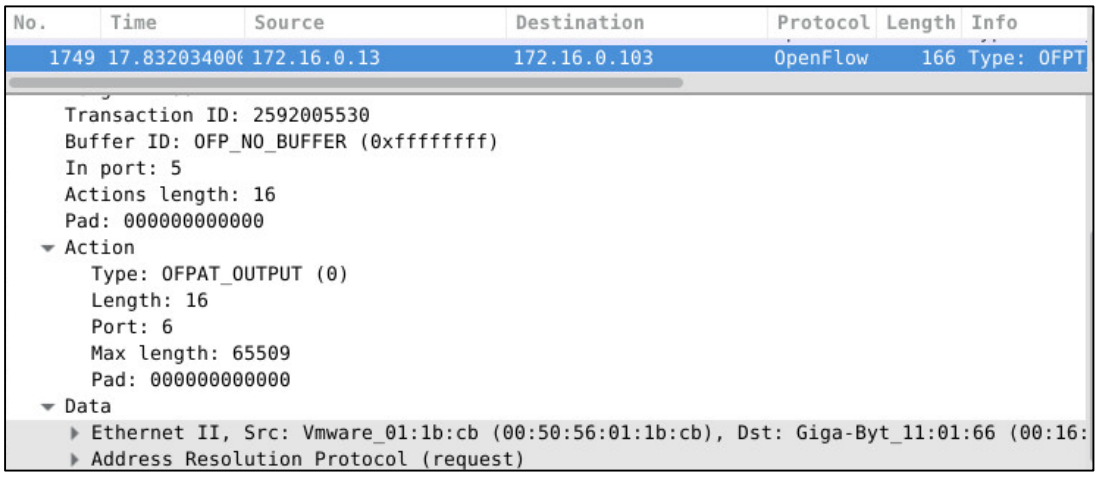
```
root@raspberrypi-1:/home/pi# ovs-vsctl set-controller OF-switch
tcp:172.16.0.10:6633
```

Kytkimille SW1-SW7 tehtiin vastaava komento. Kun tämän jälkeen testattiin pingata Host4:ltä RGCEn hosteja, huomattiin RYUn ohjaavan liikenteen onnistuneesti perille. Kuviossa 45 näkyy todennus onnistuneesta pingistä host4:n ja host1:n välillä.

```
administrator@SP-Ubuntu:~$ ping 203.0.113.101
PING 203.0.113.101 (203.0.113.101) 56(84) bytes of data.
64 bytes from 203.0.113.101: icmp_seq=1 ttl=64 time=32.3 ms
64 bytes from 203.0.113.101: icmp_seq=2 ttl=64 time=27.5 ms
64 bytes from 203.0.113.101: icmp_seq=3 ttl=64 time=27.8 ms
^C
--- 203.0.113.101 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 27.594/29.256/32.338/2.190 ms
administrator@SP-Ubuntu:~$ ping 203.0.113.102
PING 203.0.113.102 (203.0.113.102) 56(84) bytes of data.
64 bytes from 203.0.113.102: icmp_seq=1 ttl=64 time=64.5 ms
64 bytes from 203.0.113.102: icmp_seq=2 ttl=64 time=26.7 ms
64 bytes from 203.0.113.102: icmp_seq=3 ttl=64 time=30.2 ms
^C
--- 203.0.113.102 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 26.741/40.510/64.579/17.078 ms
administrator@SP-Ubuntu:~$
```

Kuvio 45. Host4 ping Host1, Host2 todennus

Kun RYUlle saapuvia OF-paketteja kaapataan Wiresharkilla, kuviossa 46 huomataan kuinka RYU lähettää SW3:lle OFPT_Packet_out-paketin, jota ONOS ei tehnyt



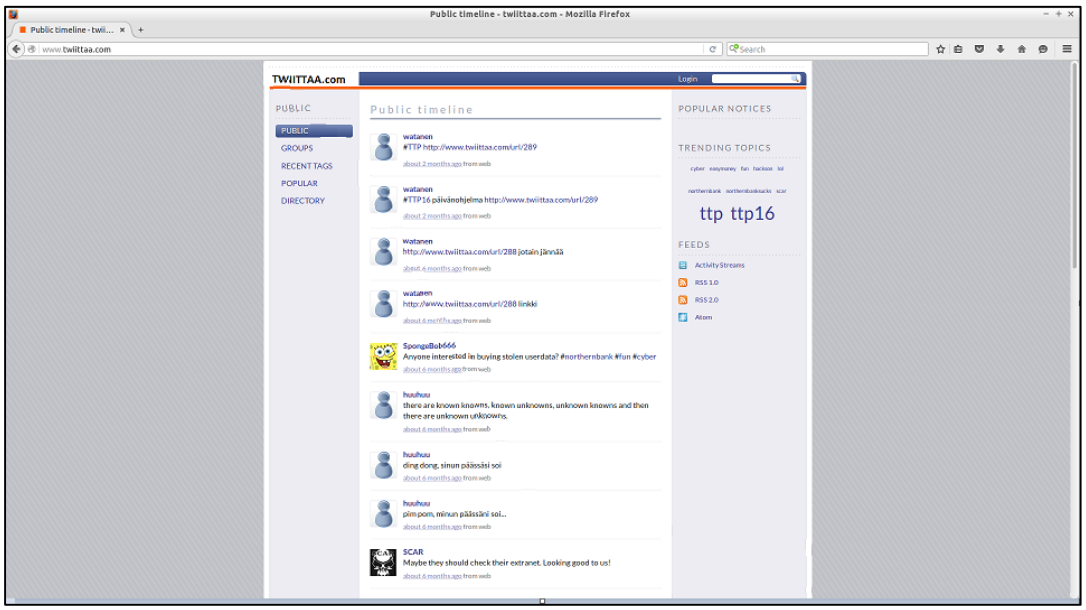
Kuvio 46. RYU OFPT_Packet_out Wireshark-kaappaus

RYU kertoo SW3:lle mitä portista 5 sisään tulevalle ARP Requestille tehdään. Kun RYU on oppinut verkon päätelaitteiden sijainnin se osaa ohjata paketin ulos portista 5.

Koska RGCE-ympäristöllä pyritään mallintamaan realistisesti oikeaa internetiä, myös sen runkoverkko koostuu useista verkko-operaattoreista, joilla on käytössä omat AS-alueensa. SDN-testbed muodostaa oman verkko-operaattorinsa, joka käyttää AS-numeroa 64000. SDN-verkko-operaattorilla on BGP-naapuruus kahden muun AS-alueen kanssa, joista se saa mainostuksia kaikista RGCEn julkisista verkoista. BGP-naapuruudet muodostetaan vRouter1:n ja vRouter2:n välityksellä, mutta niiden tarkemmat konfiguraatiot eivät kuulu tämän työn sisältöön. Mutta koska SpiderNetissä sijaitseva Host4 on nyt osa SDN-testbediä, myös sieltä on pääsy koko RGCE-ympäristöön. RGCEn infraan kuuluu mm. DNS-palvelimet. Kun Host4:lle lisätään oletusyhdyskäytäväksi vRouter1 ja sille konfiguroidaan nimipalvelin, myös sen selaimelle voi katsoa eri RGCEssä sijaitsevia verkkosivuja. Nimipalvelimeksi asetetaan 2.16.96.5, sen määrittely tehdään /etc/resolv.conf kansioon.

```
administrator@SP-ubuntu:~$ sudo ip route add default via
203.0.113.1
administrator@SP-ubuntu:~$ sudo nano /etc/resolv.conf
nameserver 2.16.96.5
```

Tämän jälkeen Host4:ltä päästin selaimella onnistuneesti RGCE:n erilaisiin palveluihin. Kuviossa 47 näkyy esimerkkinä RGCEn Twitteriä mallintavan twiittaa.com palvelun etusivu Host4:n selaimelta käsin.



Kuvio 47. Twiitta.com etusivu

10.2.2 SpiderNetin kontrolleriverkon pystytys

Viimeinen osa SDN-testbedin laajennusta oli asentaa omat SDN-kontrollerit SpiderNetissä olevalle tietokoneelle, jolloin laboratorioympäristöä voidaan ajaa paikallisesti ilman RGCE:tä. SpiderNetiin asennettiin myös kontrollereiksi ONOS ja RYU. Palvelimet asennettiin työssä aiemmin esitettyjen ohjeiden mukaisesti. Koska kaikille OVS-kytkimillä on nyt neljä eri kontrolleria, johon ne voivat ottaa yhteyden, tehtiin kontrollerin vaihtamisen helpottamiseksi jokaiselle kytkimelle neljä skriptiä, jotka ajavan kontrollerin vaihtoon tarvittavat komennot. Skripteille annettiin nimeksi rgceonos.sh, rgceryu.sh, spidernetonos.sh ja spidernetryu.sh, ja niihin annettiin suorittamisoikeudet komennolla chmod+x. Raspberry Piiden kytkimillä SW5-SW7, skripteissä määritellään ainoastaan kontrollerin IP-osoite. Rgceonos.sh on seuraavanlainen:

```
ovs-vsctl set-controller OF-switch tcp:172.16.0.10:6633
```

Rgceryu.sh on hyvin samankaltainen:

```
ovs-vsctl set-controller OF-switch tcp:172.16.0.13:6633
```

SpiderNetin kontrollereihin yhteydenotto ei eroa muuten kuin aliverkon osalta. Spidernetonos.sh on seuraavanlainen:

```
ovs-vsctl set-controller OF-switch tcp:172.16.1.10:6633
```

Spidernetryu.sh on lähes identtinen edellisen kanssa:

```
ovs-vsctl set-controller OF-switch tcp:172.16.1.13:6633
```

SW1-SW4:n osalta skriptit eroavat hieman, koska kontrollerin ollessa RGCE:ssä, OF yhteys kulkee Out-OF-Band-Management verkko pitkin, kun taas SpiderNetin kontrollereihin yhteys otetaan In-Band-Managementin kautta. Jälkimmäisessä tapauksessa OVS:n rajapinnalle pitää antaa IP-osoite 172.16.1.0/24 verkosta. RGCE:n kontrolleriin yhdistettäessä IP-osoite poistetaan rajapinnalta. Rgceonos.sh näyttää kytkimillä SW1-SW4 seuraavanlaiselta:

```
ip addr flush of-switch
ovs-vsctl set-controller of-switch tcp:172.16.0.10:6633
```

Rgceryu.sh on samankaltainen:

```
ip addr flush of-switch
ovs-vsctl set-controller of-switch tcp:172.16.0.13:6633
```

Spidernetonos.sh on jokaisella kytkimellä hieman erilainen, koska niille annetaan IP-osoite liitteen 19 mukaisesti:

```
ifconfig of-switch <ip-osoite>
ovs-vsctl set-controller of-switch tcp:172.16.1.10:6633
```

Spidernetryu.sh ei eroa tästä skrptistä kuin IP-osoitteen osalta:

```
ifconfig of-switch <ip-osoite>
ovs-vsctl set-controller of-switch tcp:172.16.1.13:6633
```

Kun spidernetryu.sh skripti on ajettu kaikilla kytkimillä, voidaan huomata, että kaikki kytkimet ovat saaneet yhteyden SpiderNetissä sijaitsevaan RYUhun. Kuviossa 50 näkyy onnistunut yhteyden muodostus SW2:lta käsin katsottuna.

```
root@sw2:~# ovs-vsctl show
2a8b692a-a028-45cd-a349-1195e9cb3792
    Bridge of-switch
        Controller "tcp:172.16.1.13:6633"
            is_connected: true
        Port "eth2"
            Interface "eth2"
        Port "eth4"
            Interface "eth4"
        Port "eth5"
            Interface "eth5"
        Port "eth1"
            Interface "eth1"
        Port "eth3"
            Interface "eth3"
        Port of-switch
            Interface of-switch
                type: internal
    ovs_version: "2.3.0"
root@sw2:~# _
```

Kuvio 48. SW2 ovs-vsctl show tuloste

Tämän jälkeen liikenne kulkee taas SpiderNetin ja RGCEn välillä. Kuviossa 51 näkyy todennus Host2:n ja Host4:n välisestä ICMP-liikenteestä.

```
root@debian:~# ping 203.0.113.104
PING 203.0.113.104 (203.0.113.104) 56(84) bytes of data.
From 203.0.113.102 icmp_seq=9 Destination Host Unreachable
From 203.0.113.102 icmp_seq=10 Destination Host Unreachable
From 203.0.113.102 icmp_seq=11 Destination Host Unreachable
64 bytes from 203.0.113.104: icmp_seq=12 ttl=128 time=45.7 ms
64 bytes from 203.0.113.104: icmp_seq=13 ttl=128 time=1.27 ms
64 bytes from 203.0.113.104: icmp_seq=14 ttl=128 time=1.56 ms
64 bytes from 203.0.113.104: icmp_seq=15 ttl=128 time=1.49 ms
^C
--- 203.0.113.104 ping statistics ---
15 packets transmitted, 4 received, +3 errors, 73% packet loss, time 14084ms
rtt min/avg/max/mdev = 1.277/12.516/45.726/19.174 ms, pipe 3
root@debian:~#
```

Kuvio 49. Host2 ping Host4 todennus

Verkossa tapahtuu pakettien menetystä ennen kuin RYU saa kerrottua kaikille kytkimille, mitä ARP-viesteille kuuluu tehdä. Näin suurta pakettihävikkiä ei normaalisti esiinny, mutta fyysisen ja virtuaalisen verkon yhdistävä infrastruktuuri aiheuttaa jostain syystä hitautta verkon toimintaan. Ajan loppumisen takia SpiderNetiin ei asennettu ONOSia tai mitään muutakaan kontrolleria RYUn lisäksi. Voidaan kuitenkin olettaa, että ONOS toimisi hyvin, jos laboratorioympäristö rajoittuu fyysisille laitteille, mutta se ei toimisi, jos RGCEtä haluttaisiin myös hyödyntää.

11 Yhteenveto

11.1 Työn tulokset

Työn toimeksiantona oli tutkia tapoja, joilla palvelun laatua voidaan toteuttaa SDN-verkossa. Toteutusvaiheessa huomattiin paljon puutteita avoimen lähdekoodin SDN-kontrollereissa. Useiden kontrollereiden dokumentointi on myös niin heikkotasoista ja vanhentunutta, että niiden käyttöönotto ja konfigurointi ovat hyvin vaikeaa. Työn palvelun laatua tutkivassa osiossa päädyttiin valitsemaan käytössä olevaksi kontrolleriksi RYU. Palvelun laatu mittauksissa todettiin, että RYUlla ja OVS:llä pystytään toteuttamaan yksinkertaista palvelun laatua leikkaamalla tietyn kaistan ylittävää liikennettä, mutta hienojakoisemmissa palvelunlaatuasetuksissa OVS:n linux-htb jonotusalgoritmin implementaatio ei toimi halutulla tavalla. Toinen osa työn toteutusta oli SDN-testbedin laajennus fyysisillä laitteilla. Verkon fyysinen laajennus Raspberry Piillä onnistui ja sillä pystytään jo nyt tekemään SDN-aiheisia laboratorioharjoituksia hyödyntämällä, joko pelkkää fyysistä verkkoa tai fyysisen verkon ja RGCE:n yhdistelmää. Fyysinen verkko toimii pohjana tuleville laajennuksille, joilla SDN-laboratorioverkko saadaan osaksi SpiderNet-ympäristöä. SDN-testbedissä oli käytössä kaksi SDN-

kontrolleria: RYU ja ONOS. Molempia pystytään hyödyntämään labrakäytössä fyysisessä verkossa, mutta ONOSia ei saatu toimimaan, kun haluttiin saada liikennettä kulkemaan fyysisen ja virtuaalisen ympäristön välillä.

11.2 Kehityskohteet

Työn toteutus tarjoaa useita jatkokehitysmahdollisuuksia. Palvelunlaatumittauksia voitaisiin jatkaa erilaisilla DOS-hyökkäyksillä ja tutkia esimerkiksi millaisilla liikennemäärillä verkon kontrollerin suorituskyky tulee rajoittavaksi tekijäksi uusia vuosääntöjä lisätessä. Sekä asiakasliikenteen, että kontrolliliikenteen salaus voitaisiin toteuttaa testiympäristöön ja tutkia kuinka paljon se lisää viivettä. Tämä vaatisi PKI-infrastruktuurin suunnittelun ja implementoimisen ympäristöö. Avoimen lähdekoodin kontrollereista löytyviä puutteita tai bugeja voitaisiin korjata tai ohjelmoida uusia applikaatioita palvelun laadun edistämiseksi. Esimerkiksi graafinen käyttöliittymä, jolla voitaisiin määrittää kahden päätepisteen välille tietyille IP-osoitteilla tai porteille korkeampi prioriteetti, jonka jälkeen kontrolleri lisää tarvittavat vuot kaikille verkon laitteille ja tekisi dynaamisesti muutoksia verkkotopologian muuttuessa.

Kuviossa 39 on esitettynä tämän hetkinen SpiderNet-verkon topologia. Työryhmien verkkojen välillä on eri laite valmistajien runkoverkkoja. Uutena runkoverkkona voitaisiin toteuttaa SDN-Core, joka koostuisi Raspberry Piistä ja jota voitaisiin hyödyntää SDN-aiheisissa laboratorioharjoituksissa. SDN-Corella voitaisiin tutkia erilaisia runkoverkossa käytettäviä protokollia kuten MPLS, Q-in-Q, Mac-in-Mac ja testata toimivatko OVS:n implementaatiot näille protokollille toimivat yhteen muiden laitevalmistajien verkkolaitteiden kanssa ja millä kontrollereilla näiden protokollien konfigurointi onnistuu.

11.3 Pohdinta

Kokonaisuudessaan opinnäytetyön tekeminen oli hyödyllinen ja opettavainen prosessi sekä tekijälle että tilaajalle. Teoreettisen tiedon kartuttamisen lisäksi työn aikana tuli valtavasti käytännön tietotaitoa SDN-tekniikoista ja ohjelmistoista. Työ oli myös opettavainen kokemus suunnittelun tärkeydestä tutkimusta tehdessä ja useista ongelmakohdista olisi selvitty huomattavasti nopeammin, jos työn toteutuksen suunnitteluun olisi käytetty enemmän aikaa. Työn toteutus antaa projektille ja laajemminkin JAMKille hyvän pohjan jalostaa eteenpäin SDN-tutkimus -ja koulutustyötä SDN-testbedin laajennuksen myötä. Työn raportointi pyrittiin tekemään mahdollisimman yksityiskohtaisesti, jolloin tulevat projektin työntekijät ja opinnäytetyön tekijät saavat hyvät valmiudet ympäristön jatkokehittämiseen.

Lähteet

Almquist, P. 1992. RFC 1349. Type of Service in the Internet Protocol Suite. Viitattu 1.3.2016. Https://tools.ietf.org/html/rfc1349

Andy. 2008. Marking DSCP values with Policy-Based Routing. Blogikirjoitus. Viitattu 1.3.2016. Https://cisconinja.wordpress.com/2008/11/26/marking-dscp-values-with-policy-based-routing/

Appenzeller, G., Balland, P., Barker, C., Beckmann, C., Casado, M., Crabbe, E., Cohn, D., Curtis, S., Das, S., Ding, W., Dheureuse, N., Dunbar, L., Gandham, S. Erickson, D., Gibb, G., Heller, B., Kobayashi, M., Lajos Kis, Z., Lantz, B., Madabushi, R., Malek, D., McDysan, D., McKeown, N., Mizrahi, T., Moses, Y., Nygren, A., Orr, M., Pettit, J., Pfaff, B., Poutievski, L., Price, R., Ramanathan, R., Schneider, F., Sherwood, R., Talayco, D., Takahashi, M., Tonsing, J., Tourrilhes, J., Vicisano, L., Ward, D., Yabe, T., Yiakoumis, Y., Yadav, N. & Yap, KK. 2014. OF Switch Specification 1.5.0. OpenFlow-kytkin standardi. Viitattu 29.2.2016 Https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/OF/OF-switch-v1.5.0.noipr.pdf

Arindam, P. 2013. QoS in Data Networks: Protocols and Standards. PDF-tiedosto. Viitattu 29.2.2016. Http://www.cse.wustl.edu/~jain/cis788-99/ftp/qos_protocols.pdf

Availabilty Measurement. 2004. Cisco Systems. Cisco esitysmateriaali saatavuuden mittauksesta. Viitattu 2.3.2016. Http://www.cisco.com/networkers/nw04/presos/docs/NMS-2201.pdf

Baker, F., Black, D., Blake, S. & Nichols, K. 1998. RFC 2474. Definition OF the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. Viitattu 29.2.2016. Https://tools.ietf.org/html/rfc2474

Balchunas, A. 2010. QoS and Queuing. Blogikirjoitus. Viitattu 17.3.2016. Http://www.routeralley.com/guides/qos_queuing.pdf

Benitez, J., Bugenhagen, M., Chiosi, M., Clarke, D., Cui, C., Damker, H., Delisle, D., Demaria, E., Deng, H., Fargano, M., Feger, J., Fukui, M., Guardini, I., Khan, W., Kolias, C., Lopez, D., Loudier, Q., Matsuazaki, T., Manzalini, A., Michel, U., Minerva, R., Ogaki, K., Reid, A., Ruhl, F., Salguero, F., Sen, P., Shimano, K. & Willis P. 2012. Network Functions Virtualisation. White Paper. Viitattu 9.3.2016. Https://portal.etsi.org/NFV/NFV_White_Paper.pdf

Chapman, Joseph. 2009. Deploying QoS for Cisco IP and next-generation networks.

Denazis, S., Hadi Salim, J., Haleplidis, E., Koufopavlou, O., Meyer, D. & Pentikousis, K. 2015. RFC 7426 SOFtware-Defined Networking (SDN): Layers and Architecture Terminology. Viitattu 23.2.2016 Https://tools.ietf.org/html/rfc7426

Devera, M. 2002a. HTB Linux queuing discipline manual. Linux-htb käyttöopas. Viitattu 14.4.2016. Http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm

Devera, M. 2002b. Hierarchial token bucket theory. Linux-htb käyttöopas. Viitattu 14.4.2016. Http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm

Ghosh, A. 2014. Why Software Defined Data Center is Emerging in IT. Verkkoartikkeli. Viitattu 23.2.2016. Https://thecustomizewindows.com/2014/02/why-sOFtware-defined-data-center-is-emerging-in-it/

JDSU. 2013. T-Berd/MTS-6000A tuote-esittely. Viitattu 22.4.2016. Http://www.viavi-solutions.com/sites/default/files/technical-library-files/TB-6000a-ds-tfs-tm-ae.pdf

Hasib, M., Schormans, J. 2003. Limitations of Passive & Active Measurement Methods in Packet Networks. Verkkoartikkeli palvelun laatu mittauksista. Viitattu 16.3.2016. Http://www.ee.ucl.ac.uk/lcs/previous/LCS2003/113.pdf

Kumar, P. 2012. Understanding IP Precedence, TOS & DSCP. Blogikirjoitus. Viitattu 1.3.2016. Https://blogs.manageengine.com/network/netflowanalyzer/2012/04/24/understanding-ip-precedence-tos-dscp.html

OF. n.d.a. Flowgrammable. OpenFlow-protokollan esittely. Viitattu 29.2.2016, Http://flowgrammable.org/sdn/OF/

OF. n.d.b. Open Networking Foundation. OpenFlow-protokollan esittely. Viitattu 23.2.2016. Https://www.opennetworking.org/sdn-resources/OF

onosproject. n.d. onosprojektin verkkosivut. Viitattu 15.5.2016. Http://onosproject.org/

Rasbian. n.d. Rasbian-käyttöjärjestelmän esittely. Viitattu 16.3.2016. Https://www.raspberrypi.org/documentation/raspbian/

RGCE. n.d. JYVSECTECin verkkosivut. Viitattu 2.5.2016. Http://jyvsectec.fi/fi/kyberymparisto/

Savola, R. 2014. DIGILE CyberTrust and Related Security Projects. Cyber Trust -projektin esittelymateriaali. Viitattu 5.2.2016. Http://www.vtt.fi/files/sites/eemeli18/12_Reijo_Savola_security_calls.pdf

Software-Defined Networking: The New Norm for Networks. ONF:n julkaisu SDN-verkoista. 2012. Viitattu 5.2.2016. Https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

SpiderNet. 2009. SpiderNet-ympäristön esittely. Viitattu 2.5.2015. Http://student.labranet.jamk.fi/SpiderNet/

Stallings, W. 2015. Foundations OF Modern Networking.

Tutustu JAMKiin. n.d. JAMKin kotisivut. Viitattu 22.2.2016. Http://www.jamk.fi/fi/Tietoa-JAMKista/Tutustu-JAMKiin/

Vainionkulma-Immonen, O. 2015. activityblog DIGILE. Blogikirjoitus. Viitattu 5.2.2016. Http://www.activityblog.fi/tag/cyber-trust/

VyOS. 2016. VyOSin wiki-sivut. Viitattu 19.4.2016. Http://vyos.net/wiki/Main_Page

What are SDN Controllers (or SDN Controllers Platforms)?. n.d. Verkkoartikkeli. Viitattu 7.3.2016. Https://www.sdxcentral.com/resources/sdn/sdn-controllers/

What is a Raspberry Pi? n.d. Raspberry Piin esittely. Viitattu 16.3.2016. Https://www.raspberrypi.org/help/what-is-a-raspberry-pi/

What is ONF?. 2013. ONF:n esittely. Viitattu 5.2.2016. Https://www.opennetworking.org/images/stories/downloads/about/onf-what-why.pdf

What is Open vSwitch?. n.d. Open vSwitchin manuaali. Viitattu 16.3.2016. Https://github.com/openvswitch/ovs/blob/master/README.md

What's RYU? n.d. RYUn esittely. Viitattu 11.4.2016. Https://osrg.github.io/ryu/

Liitteet

Liite 1. VyOSin asennus

```
vyos@vyos:~$ install image
Welcome to the VyOS install program.  This script
will walk you through the process of installing the
VyOS image to a local hard drive.
Would you like to continue? (Yes/No) [Yes]: [return]
Probing drives: OK
Looking for pre-existing RAID groups...none found.
The VyOS image will require a minimum 1000MB root.
Would you like me to try to partition a drive automatically
or would you rather partition it manually with parted?  If
you have already setup your partitions, you may skip this step

Partition (Auto/Parted/Skip) [Auto]: [return]

I found the following drives on your system:
 sda  2147MB


Install the image on? [sda]: [return]

This will destroy all data on /dev/sda.
Continue? (Yes/No) [No]: Yes

How big of a root partition should I create? (1000MB - 2147MB) [2147]MB: [return]

Creating filesystem on /dev/sda1: OK
Done!
Mounting /dev/sda1...
What would you like to name this image? [VyOS_999.hydrogen.11291501]:
OK.  This image will be named: VyOS_999.hydrogen.11291501
Copying squashfs image...
Copying kernel and initrd images...
Done!
I found the following configuration files:
    /config/config.boot
    /opt/vyatta/etc/config.boot.default
Which one should I copy to sda? [/config/config.boot]: [return]

Copying /config/config.boot to sda.
Enter password for administrator account
Enter password for user 'vyos': <removed>
Retype password for user 'vyos': <removed>
I need to install the GRUB boot loader.
I found the following drives on your system:
 sda  2147MB


Which drive should GRUB modify the boot partition on? [sda]: [return]

Setting up grub: OK
Done!
vyos@vyos:~$
```

Liite 2. Raspberrypi-2 konfiguraatiot palvelun laatumittaukset

```
root@raspberrypi-2:/home/pi# ip link add veth1 type veth peer name veth2
root@raspberrypi-2:/home/pi# ip link add veth3 type veth peer name veth4
root@raspberrypi-2:/home/pi# ip link add veth5 type veth peer name veth6
root@raspberrypi-2:/home/pi# ip link add veth7 type veth peer name veth8
root@raspberrypi-2:/home/pi# ovs-vsctl add-br ovs1
root@raspberrypi-2:/home/pi# ovs-vsctl add-br ovs2
root@raspberrypi-2:/home/pi# ovs-vsctl add-br ovs3
root@raspberrypi-2:/home/pi# ovs-vsctl add-br ovs4
root@raspberrypi-2:/home/pi# ovs-vsctl add-port ovs1 veth1
```

```
root@raspberrypi-2:/home/pi# ovs-vsctl add-port ovs1 veth8
root@raspberrypi-2:/home/pi# ovs-vsctl add-port ovs2 eth3
root@raspberrypi-2:/home/pi# ovs-vsctl add-port ovs2 veth2
root@raspberrypi-2:/home/pi# ovs-vsctl add-port ovs2 veth3
root@raspberrypi-2:/home/pi# ovs-vsctl add-port ovs3 eth3
root@raspberrypi-2:/home/pi# ovs-vsctl add-port ovs3 veth4
root@raspberrypi-2:/home/pi# ovs-vsctl add-port ovs3 veth5
root@raspberrypi-2:/home/pi# ovs-vsctl add-port ovs4 eth4
root@raspberrypi-2:/home/pi# ovs-vsctl add-port ovs4 veth6
root@raspberrypi-2:/home/pi# ovs-vsctl add-port ovs4 veth7
root@raspberrypi-2:/home/pi# ovs-vsctl set bridge ovs1 protocols=OpenFlow13
root@raspberrypi-2:/home/pi# ovs-vsctl set bridge ovs2 protocols=OpenFlow13
root@raspberrypi-2:/home/pi# ovs-vsctl set bridge ovs3 protocols=OpenFlow13
root@raspberrypi-2:/home/pi# ovs-vsctl set bridge ovs4 protocols=OpenFlow13
root@raspberrypi-2:/home/pi# ovs-vsctl set-controller ovs1
tcp:172.16.0.11:6633
root@raspberrypi-2:/home/pi# ovs-vsctl set-controller ovs2
tcp:172.16.0.11:6633
root@raspberrypi-2:/home/pi# ovs-vsctl set-controller ovs3
tcp:172.16.0.11:6633
root@raspberrypi-2:/home/pi# ovs-vsctl set-controller ovs4
tcp:172.16.0.11:6633
```

Liite 3. OVS1-vuotaulu

```
root@raspberrypi-2:/home/pi# ovs-ofctl -O openflow13 dump-flows ovs1
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=254.990s, table=0, n_packets=508, n_bytes=25908, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=254.989s, table=0, n_packets=4345, n_bytes=548806, priority=0 actions=goto_table:1
 cookie=0x0, duration=254.453s, table=1, n_packets=2117, n_bytes=273302, priority=1,in_port=1,dl_dst=b8:27:eb:ea:06:6d actions=output:2
 cookie=0x0, duration=254.416s, table=1, n_packets=2120, n_bytes=251865, priority=1,in_port=2,dl_dst=08:00:27:ea:f2:20 actions=output:1
 cookie=0x0, duration=61.850s, table=1, n_packets=8, n_bytes=594, priority=1,in_port=1,dl_dst=00:80:16:8a:3c:e9 actions=output:2
 cookie=0x0, duration=61.778s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=2,dl_dst=00:80:16:8c:16:5b actions=output:1
 cookie=0x0, duration=58.326s, table=1, n_packets=1, n_bytes=60, priority=1,in_port=2,dl_dst=00:80:16:8c:0d:fa actions=output:1
 cookie=0x0, duration=55.129s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=2,dl_dst=08:00:27:de:21:80 actions=output:1
 cookie=0x0, duration=43.428s, table=1, n_packets=13, n_bytes=2722, priority=1,in_port=2,dl_dst=9c:eb:e8:06:10:2c actions=output:1
 cookie=0x0, duration=255.019s, table=1, n_packets=82, n_bytes=20023, priority=0 actions=CONTROLLER:65535
root@raspberrypi-2:/home/pi#
```

Liite 4. OVS2-vuotaulu

```
root@raspberrypi-2:/home/pi# ovs-ofctl -O openflow13 dump-flows ovs2
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=275.084s, table=0, n_packets=548, n_bytes=27948, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=275.083s, table=0, n_packets=4659, n_bytes=587838, priority=0 actions=goto_table:1
 cookie=0x0, duration=274.559s, table=1, n_packets=2273, n_bytes=293046, priority=1,in_port=1,dl_dst=b8:27:eb:ea:06:6d actions=output:3
 cookie=0x0, duration=274.502s, table=1, n_packets=2277, n_bytes=270149, priority=1,in_port=3,dl_dst=08:00:27:ea:f2:20 actions=output:1
 cookie=0x0, duration=81.960s, table=1, n_packets=3, n_bytes=218, priority=1,in_port=2,dl_dst=00:80:16:8a:3c:e9 actions=output:3
 cookie=0x0, duration=81.864s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=3,dl_dst=00:80:16:8c:16:5b actions=output:2
 cookie=0x0, duration=78.411s, table=1, n_packets=1, n_bytes=60, priority=1,in_port=3,dl_dst=00:80:16:8c:0d:fa actions=output:4
 cookie=0x0, duration=78.378s, table=1, n_packets=0, n_bytes=0, priority=1,in_port=4,dl_dst=00:80:16:8a:3c:e9 actions=output:3
 cookie=0x0, duration=75.254s, table=1, n_packets=3, n_bytes=218, priority=1,in_port=1,dl_dst=00:80:16:8a:3c:e9 actions=output:3
 cookie=0x0, duration=75.214s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=3,dl_dst=08:00:27:de:21:80 actions=output:1
 cookie=0x0, duration=63.515s, table=1, n_packets=13, n_bytes=2722, priority=1,in_port=3,dl_dst=9c:eb:e8:06:10:2c actions=output:1
 cookie=0x0, duration=275.114s, table=1, n_packets=85, n_bytes=21185, priority=0 actions=CONTROLLER:65535
root@raspberrypi-2:/home/pi#
```

Liite 5. OVS3-vuotaulu

```
root@raspberrypi-2:/home/pi# ovs-ofctl -O openflow13 dump-flows ovs3
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=296.767s, table=0, n_packets=591, n_bytes=32796, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=296.766s, table=0, n_packets=5009, n_bytes=631313, priority=0 actions=goto_table:1
 cookie=0x0, duration=296.219s, table=1, n_packets=2444, n_bytes=314827, priority=1,in_port=3,dl_dst=b8:27:eb:ea:06:6d actions=output:1
 cookie=0x0, duration=296.202s, table=1, n_packets=2453, n_bytes=290691, priority=1,in_port=1,dl_dst=08:00:27:ea:f2:20 actions=output:3
 cookie=0x0, duration=103.612s, table=1, n_packets=8, n_bytes=594, priority=1,in_port=3,dl_dst=00:80:16:8a:3c:e9 actions=output:1
 cookie=0x0, duration=103.562s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=1,dl_dst=00:80:16:8c:16:5b actions=output:3
 cookie=0x0, duration=100.121s, table=1, n_packets=0, n_bytes=0, priority=1,in_port=1,dl_dst=00:80:16:8c:0d:fa actions=output:3
 cookie=0x0, duration=96.914s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=1,dl_dst=08:00:27:de:21:80 actions=output:3
 cookie=0x0, duration=85.212s, table=1, n_packets=13, n_bytes=2722, priority=1,in_port=1,dl_dst=9c:eb:e8:06:10:2c actions=output:3
 cookie=0x0, duration=296.796s, table=1, n_packets=87, n_bytes=22239, priority=0 actions=CONTROLLER:65535
root@raspberrypi-2:/home/pi#
```

Liite 6. OVS4-vuotaulu

```
root@raspberrypi-2:/home/pi# ovs-ofctl -O openflow13 dump-flows ovs4
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=314.413s, table=0, n_packets=313, n_bytes=15963, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=314.412s, table=0, n_packets=81, n_bytes=23017, priority=0 actions=goto_table:1
 cookie=0x0, duration=117.728s, table=1, n_packets=1, n_bytes=60, priority=1,in_port=1,dl_dst=00:80:16:8c:0d:fa actions=output:3
 cookie=0x0, duration=117.712s, table=1, n_packets=0, n_bytes=0, priority=1,in_port=3,dl_dst=00:80:16:8a:3c:e9 actions=output:1
 cookie=0x0, duration=314.443s, table=1, n_packets=80, n_bytes=22957, priority=0 actions=CONTROLLER:65535
root@raspberrypi-2:/home/pi#
```

Liite 7. OVS5-vuotaulu

```
root@raspberrypi-1:/home/pi# ovs-ofctl -O openflow13 dump-flows ovs5
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=212.431s, table=0, n_packets=424, n_bytes=23532, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=212.417s, table=0, n_packets=130, n_bytes=22185, priority=0 actions=goto_table:1
 cookie=0x0, duration=104.730s, table=1, n_packets=26, n_bytes=4100, priority=1,in_port=LOCAL,dl_dst=9c:eb:e8:06:10:2c actions=output:1
 cookie=0x0, duration=104.709s, table=1, n_packets=44, n_bytes=3632, priority=1,in_port=1,dl_dst=b8:27:eb:ea:06:6d actions=LOCAL
 cookie=0x0, duration=19.874s, table=1, n_packets=7, n_bytes=534, priority=1,in_port=1,dl_dst=00:80:16:8a:3c:e9 actions=output:2
 cookie=0x0, duration=19.838s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=2,dl_dst=00:80:16:8c:16:5b actions=output:1
 cookie=0x0, duration=16.400s, table=1, n_packets=1, n_bytes=60, priority=1,in_port=2,dl_dst=00:80:16:8c:0d:fa actions=output:1
 cookie=0x0, duration=13.190s, table=1, n_packets=2, n_bytes=120, priority=1,in_port=2,dl_dst=08:00:27:de:21:80 actions=output:1
 cookie=0x0, duration=212.437s, table=1, n_packets=48, n_bytes=13619, priority=0 actions=CONTROLLER:65535
root@raspberrypi-1:/home/pi#
```

Liite 8. OVS7-vuotaulu

```
root@raspberrypi-1:/home/pi# ovs-ofctl -O openflow13 dump-flows ovs7
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=112.183s, table=0, n_packets=0, n_bytes=0, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=112.183s, table=0, n_packets=0, n_bytes=0, priority=0 actions=goto_table:1
 cookie=0x0, duration=112.194s, table=1, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
root@raspberrypi-1:/home/pi#
```

Liite 9. Liikennevirta2 ennen merkkausta

```
▾ Internet Protocol Version 4, Src: 203.0.113.102 (203.0.113.102), Dst: 203.0.113.105 (203.0.113.105)
    Version: 4
    Header Length: 20 bytes
  ▸ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 512
    Identification: 0x0000 (0)
  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: Unknown (254)
  ▸ Header checksum: 0xbf2f [validation disabled]
    Source: 203.0.113.102 (203.0.113.102)
    Destination: 203.0.113.105 (203.0.113.105)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▸ Data (492 bytes)
```

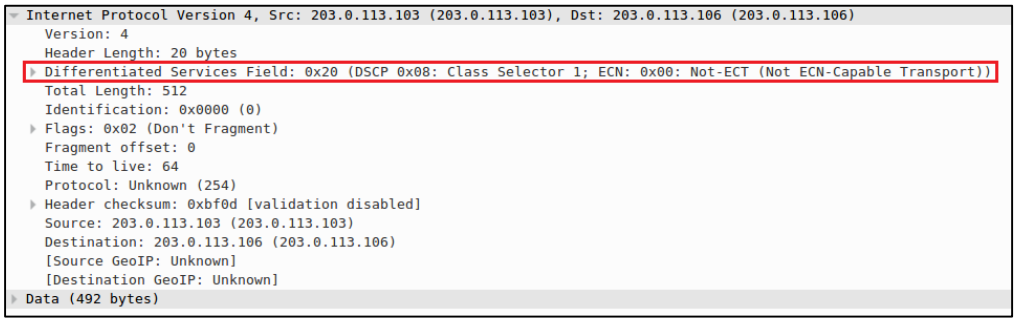
Liite 10. Liikennevirta2 merkkauksen jälkeen

```
▾ Internet Protocol Version 4, Src: 203.0.113.102 (203.0.113.102), Dst: 203.0.113.105 (203.0.113.105)
    Version: 4
    Header Length: 20 bytes
  ▸ Differentiated Services Field: 0x40 (DSCP 0x10: Class Selector 2; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 512
    Identification: 0x0000 (0)
  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: Unknown (254)
  ▸ Header checksum: 0xbeef [validation disabled]
    Source: 203.0.113.102 (203.0.113.102)
    Destination: 203.0.113.105 (203.0.113.105)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▸ Data (492 bytes)
```

Liite 11. Liikennevirta3 ennen merkkausta

```
▾ Internet Protocol Version 4, Src: 203.0.113.103 (203.0.113.103), Dst: 203.0.113.106 (203.0.113.106)
    Version: 4
    Header Length: 20 bytes
  ▸ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 512
    Identification: 0x0000 (0)
  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: Unknown (254)
  ▸ Header checksum: 0xbf2d [validation disabled]
    Source: 203.0.113.103 (203.0.113.103)
    Destination: 203.0.113.106 (203.0.113.106)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▸ Data (492 bytes)
```

Liite 12. Liikennevirta3 merkkauksen jälkeen



Liite 13. Mittaus 1 JDSU-raportti

Results: Stream 1 / L2 Link Results	
Total Util %, Average	0.047
Total Util %, Current	0.048
Total Util %, Minimum	0.034
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	0.48
Rx Mbps, Cur L2	0.47
Tx Mbps, Cur L1	2.03
Tx Mbps, Cur L2	2.00

Round Trip Delay (us), Average	1,441.38
Round Trip Delay (us), Current	1,486.31
Round Trip Delay (us), Minimum	1,271.95
Round Trip Delay (us), Maximum	2,021.52
Packet Jitter (us), Average	98.30
Packet Jitter (us), Max Average	131.07
Packet Jitter (us), Peak	534.53
Packet Jitter (us), Instantaneous	190.46
Received Frames	2,330
Transmitted Frames	10,064
Tx Acterna Frames	10,064
Rx Acterna Frames	2,330
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	3,536,940
Tx Frame Bytes	15,277,152

Results: Stream 2 / L2 Link Results	
Total Util %, Average	0.851
Total Util %, Current	0.858
Total Util %, Minimum	0.819
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	8.58
Rx Mbps, Cur L2	8.46
Tx Mbps, Cur L1	20.26
Tx Mbps, Cur L2	20.00
Round Trip Delay (us), Average	1,501.83
Round Trip Delay (us), Current	1,463.53
Round Trip Delay (us), Minimum	1,288.34
Round Trip Delay (us), Maximum	2,379.92
Packet Jitter (us), Average	65.54
Packet Jitter (us), Max Average	163.84
Packet Jitter (us), Peak	817.15
Packet Jitter (us), Instantaneous	6.14
Received Frames	42,208
Transmitted Frames	100,444
Tx Acterna Frames	100,444
Rx Acterna Frames	42,208
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	64,071,744
Tx Frame Bytes	152,473,992

Liite 14. Mittaus 2 JDSU-raportti

Results: Stream 1 / L2 Link Results	
Total Util %, Average	0.152
Total Util %, Current	0.153
Total Util %, Minimum	0.151
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	1.53
Rx Mbps, Cur L2	1.51
Tx Mbps, Cur L1	2.03
Tx Mbps, Cur L2	2.00

Round Trip Delay (us), Average	6,599,510.00
Round Trip Delay (us), Current	8,073,530.00
Round Trip Delay (us), Minimum	1,398,260.00
Round Trip Delay (us), Maximum	8,076,260.00
Packet Jitter (us), Average	2,654.21
Packet Jitter (us), Max Average	2,818.05
Packet Jitter (us), Peak	4,524.03
Packet Jitter (us), Instantaneous	1,998.85
Received Frames	7,555
Transmitted Frames	10,065
Tx Acterna Frames	10,065
Rx Acterna Frames	7,555
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	11,468,490
Tx Frame Bytes	15,278,670

Results: Stream 2 / L2 Link Results	
Total Util %, Average	0.406
Total Util %, Current	0.406
Total Util %, Minimum	0.406
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	4.06
Rx Mbps, Cur L2	4.01
Tx Mbps, Cur L1	20.26
Tx Mbps, Cur L2	20.00
Round Trip Delay (us), Average	3,029,150.00
Round Trip Delay (us), Current	3,029,030.00

Round Trip Delay (us), Minimum	3,025,870.00
Round Trip Delay (us), Maximum	3,029,890.00
Packet Jitter (us), Average	32.77
Packet Jitter (us), Max Average	360.45
Packet Jitter (us), Peak	3,155.97
Packet Jitter (us), Instantaneous	10.24
Received Frames	20,145
Transmitted Frames	100,444
Tx Acterna Frames	100,444
Rx Acterna Frames	20,145
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	30,580,110
Tx Frame Bytes	152,473,992

Liite 15. Mittaus 3a JDSU-raportti

Results: Stream 1 / L2 Link Results	
Total Util %, Average	0.047
Total Util %, Current	0.054
Total Util %, Minimum	0.034
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	0.54
Rx Mbps, Cur L2	0.53
Tx Mbps, Cur L1	2.03
Tx Mbps, Cur L2	2.00

Round Trip Delay (us), Average	1,429.67
Round Trip Delay (us), Current	1,475.93
Round Trip Delay (us), Minimum	1,259.66
Round Trip Delay (us), Maximum	1,826.96
Packet Jitter (us), Average	98.30
Packet Jitter (us), Max Average	131.07
Packet Jitter (us), Peak	444.42
Packet Jitter (us), Instantaneous	192.51
Received Frames	2,344
Transmitted Frames	10,064
Tx Acterna Frames	10,064
Rx Acterna Frames	2,344
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	3,558,192
Tx Frame Bytes	15,277,152

Results: Stream 2 / L2 Link Results	
Total Util %, Average	0.852
Total Util %, Current	0.851
Total Util %, Minimum	0.826
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	8.51
Rx Mbps, Cur L2	8.40
Tx Mbps, Cur L1	20.26
Tx Mbps, Cur L2	20.00
Round Trip Delay (us), Average	1,495.10
Round Trip Delay (us), Current	1,465.56

Round Trip Delay (us), Minimum	1,276.05
Round Trip Delay (us), Maximum	2,269.33
Packet Jitter (us), Average	65.54
Packet Jitter (us), Max Average	163.84
Packet Jitter (us), Peak	684.03
Packet Jitter (us), Instantaneous	206.85
Received Frames	42,232
Transmitted Frames	100,443
Tx Acterna Frames	100,443
Rx Acterna Frames	42,232
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	64,108,176
Tx Frame Bytes	152,472,474

Liite 16. Mittaus 3b JDSU-raportti

Results: Stream 1 / L2 Link Results	
Total Util %, Average	0.130
Total Util %, Current	0.145
Total Util %, Minimum	0.091
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	1.45
Rx Mbps, Cur L2	1.43
Tx Mbps, Cur L1	2.02
Tx Mbps, Cur L2	1.99

Round Trip Delay (us), Average	1,495.45
Round Trip Delay (us), Current	1,597.17
Round Trip Delay (us), Minimum	1,288.34
Round Trip Delay (us), Maximum	2,400.40
Packet Jitter (us), Average	98.30
Packet Jitter (us), Max Average	163.84
Packet Jitter (us), Peak	667.65
Packet Jitter (us), Instantaneous	45.06
Received Frames	6,448
Transmitted Frames	10,064
Tx Acterna Frames	10,064
Rx Acterna Frames	6,448
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	9,788,064
Tx Frame Bytes	15,277,152

Results: Stream 2 / L2 Link Results	
Total Util %, Average	0.475
Total Util %, Current	0.448
Total Util %, Minimum	0.434
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	4.48
Rx Mbps, Cur L2	4.42
Tx Mbps, Cur L1	20.26
Tx Mbps, Cur L2	20.00
Round Trip Delay (us), Average	2,019,840.00
Round Trip Delay (us), Current	2,019,820.00

Round Trip Delay (us), Minimum	2,015,310.00
Round Trip Delay (us), Maximum	2,020,620.00
Packet Jitter (us), Average	262.14
Packet Jitter (us), Max Average	688.13
Packet Jitter (us), Peak	4,579.33
Packet Jitter (us), Instantaneous	114.69
Received Frames	23,568
Transmitted Frames	100,444
Tx Acterna Frames	100,444
Rx Acterna Frames	23,568
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	35,776,224
Tx Frame Bytes	152,473,992

Liite 17. Mittaus 4a JDSU-raportti

Results: Stream 1 / L2 Link Results	
Total Util %, Average	0.161
Total Util %, Current	0.157
Total Util %, Minimum	0.124
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	1.57
Rx Mbps, Cur L2	1.55
Tx Mbps, Cur L1	2.03
Tx Mbps, Cur L2	2.00

Round Trip Delay (us), Average	1,485.31
Round Trip Delay (us), Current	1,412.85
Round Trip Delay (us), Minimum	1,280.14
Round Trip Delay (us), Maximum	4,565.14
Packet Jitter (us), Average	32.77
Packet Jitter (us), Max Average	458.75
Packet Jitter (us), Peak	2,975.74
Packet Jitter (us), Instantaneous	22.53
Received Frames	7,990
Transmitted Frames	10,064
Tx Acterna Frames	10,064
Rx Acterna Frames	7,990
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	12,128,820
Tx Frame Bytes	15,277,152

Results: Stream 2 / L2 Link Results	
Total Util %, Average	0.544
Total Util %, Current	0.546
Total Util %, Minimum	0.502
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	5.46
Rx Mbps, Cur L2	5.39
Tx Mbps, Cur L1	20.26
Tx Mbps, Cur L2	20.00
Round Trip Delay (us), Average	2,019,800.00
Round Trip Delay (us), Current	2,019,690.00
Round Trip Delay (us), Minimum	2,016,090.00
Round Trip Delay (us), Maximum	2,022,190.00
Packet Jitter (us), Average	327.68
Packet Jitter (us), Max Average	622.59
Packet Jitter (us), Peak	3,250.18
Packet Jitter (us), Instantaneous	299.01
Received Frames	26,960
Transmitted Frames	100,443
Tx Acterna Frames	100,443
Rx Acterna Frames	26,960
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	40,925,280
Tx Frame Bytes	152,472,474

Liite 18. Mittaus 4b JDSU-raportti

Results: Stream 1 / L2 Link Results	
Total Util %, Average	0.162
Total Util %, Current	0.167
Total Util %, Minimum	0.123
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	1.67
Rx Mbps, Cur L2	1.65
Tx Mbps, Cur L1	2.03
Tx Mbps, Cur L2	2.00

Round Trip Delay (us), Average	1,534.18
Round Trip Delay (us), Current	1,826.22
Round Trip Delay (us), Minimum	1,274
Round Trip Delay (us), Maximum	4,561.04
Packet Jitter (us), Average	360.45
Packet Jitter (us), Max Average	524.29
Packet Jitter (us), Peak	3,059.71
Packet Jitter (us), Instantaneous	51.20
Received Frames	8,014
Transmitted Frames	10,064
Tx Acterna Frames	10,064
Rx Acterna Frames	8,014
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	12,165,252
Tx Frame Bytes	15,277,152

Results: Stream 2 / L2 Link Results	
Total Util %, Average	0.546
Total Util %, Current	0.535
Total Util %, Minimum	0.493
Frame Size, Average	1,518
Frame Size, Current	1,518
Frame Size, Minimum	1,518
Frame Size, Maximum	1,518
Rx Mbps, Cur L1	5.35
Rx Mbps, Cur L2	5.28
Tx Mbps, Cur L1	20.25
Tx Mbps, Cur L2	19.99
Round Trip Delay (us), Average	2,019,150.00
Round Trip Delay (us), Current	2,019,890.00

Round Trip Delay (us), Minimum	1,839,240.00
Round Trip Delay (us), Maximum	2,022,700.00
Packet Jitter (us), Average	327.68
Packet Jitter (us), Max Average	1,441.79
Packet Jitter (us), Peak	4,864
Packet Jitter (us), Instantaneous	1,060.86
Received Frames	27,084
Transmitted Frames	100,443
Tx Acterna Frames	100,443
Rx Acterna Frames	27,084
Rx VLAN Frames	0
Rx Q-in-Q Frames	0
Rx Frame Bytes	41,113,512
Tx Frame Bytes	152,472,474

Liite 19. SDN-testbed osoitteet

Device	Interface	IP-address	MAC-address	Description
SW1	eth0	172.16.0.101/24	00:50:56:01:1b:4e	Controller-OB-mgmt
	eth1		00:50:56:01:1b:4f	SW1-SW2 (Switch)
	eth2		00:50:56:01:1b:50	SW1-SW3 (Switch)
	of-switch	172.16.1.111/24	00:50:56:01:1b:4f	Controller-IB-Mgmt
SW2	eth0	172.16.0.102/24	00:50:56:01:1c:5d	Controller-OB-mgmt
	eth1		00:50:56:01:1c:5e	SW2-Vrouter1 (Switch)
	eth2		00:50:56:01:1c:5f	SW2-AS5617 (Switch)
	eth3		00:50:56:01:1c:60	SW1-SW2 (Switch)
	eth4		00:50:56:01:1c:61	SW2-SW4 (Switch)
	eth5		00:50:56:01:1c:62	SW2-Host1 (Switch)
	of-switch	172.16.1.112/24	00:50:56:01:1c:5e	Controller-IB-Mgmt
SW3	eth0	172.16.0.103/24	00:50:56:01:1b:39	Controller-OB-mgmt
	eth1		00:50:56:01:1b:d3	SW1-SW3 (Switch)
	eth2		00:50:56:01:1b:d4	SW3-SW4 (Switch)
	eth3		00:50:56:01:1b:d5	SW3-FW (Switch)
	eth4		00:50:56:01:1b:d6	SW3-WebSRV (Switch)
	eth5		00:50:56:01:1b:d7	SW3-Host2 (Switch)
	of-switch	172.16.1.113/24	00:50:56:01:1b:d3	Controller-IB-Mgmt
SW4	eth0	172.16.0.104/24	00:50:56:01:1b:3d	Controller-OB-mgmt
	eth1		00:50:56:01:1b:3e	SW2-SW4 (Switch)
	eth2		00:50:56:01:1b:3f	SW3-SW4 (Switch)
	eth3		00:50:56:01:1b:d9	SW4-Host3 (Switch)
	eth4		00:50:56:01:1c:02	SW4-Vrouter2 (Switch)
	eth5		00:50:56:01:1c:5c	SW4-AS6849 (Switch)
	of-switch	172.16.1.114/24	00:50:56:01:1b:3e	Controller-IB-Mgmt
SW5	eth0			SW5-SW3
	eth1			SW5-SW6
	of-switch	172.16.1.101/24		Controller-IB-Mgmt
SW6	eth0			SW6-SW5
	eth1			SW6-SW7
	of-switch	172.16.1.102/24		Controller-IB-Mgmt
SW7	eth0			SW7-SW6
	eth1			SW7-SpiderNet-Controller
	eth2			SW7-Host4
	of-switch	172.16.1.103/24		Controller-IB-Mgmt
vRouter1	eth0	203.0.113.1/24	00:50:56:01:1b:07	SW2-Vrouter1
		217.97.216.2/24		
		172.16.1.1/24		
	eth9	172.16.0.1/24	00:50:56:01:1b:10	Controller-OB-mgmt
vRouter2	eth0	203.0.113.2/24	00:50:56:01:1b:f7	SW4-Vrouter2
		195.242.218.2/23		
	eth9	172.16.0.2/24	00:50:56:01:1c:01	Controller-OB-mgmt
RGCE-ONOS	eth0	172.16.0.10/24	00:50:56:01:1b:b0	Controller-OB-mgmt
RGCE-RYU	eth0	172.16.0.13/24	00:50:56:01:1b:b1	Controller-OB-mgmt
SpiderNet-ONOS	eth0	172.16.1.10/24		SW7-SpiderNet-Controller
SpiderNet-RYU	eth0	172.16.1.13/24		SW7-SpiderNet-Controller
Management-VM	eth0	172.16.0.100/24	00:50:56:01:1b:af	Controller-OB-mgmt
Host1	eth0	203.0.113.101/24	00:50:56:01:1b:34	SW2-Host1
Host2	eth0	203.0.113.102/24	00:50:56:01:1b:bb	SW3-Host2
Host3	eth0	203.0.113.103/24	00:50:56:01:1b:bc	SW4-Host3
Host4	eth0	203.0.113.104/24		SW7-Host4
Firewall	eth0	203.0.113.199/24	00:50:56:01:1b:cb	SW3-FW
Web-server1	eth0	203.0.113.100/24	00:50:56:01:1b:26	SW3-WebSRV

Liite 20. Raspberrypi2-konfiguraatiot SDN-testbed

```
root@raspberrypi-1:/home/pi# ovs-vsctl add-br of-switch
root@raspberrypi-1:/home/pi# ovs-vsctl add-port of-switch eth0
root@raspberrypi-1:/home/pi# ovs-vsctl add-port of-switch eth1
root@raspberrypi-1:/home/pi# ifconfig of-switch 172.16.1.102/24
root@raspberrypi-1:/home/pi# update-rc.d openvswitch-switch defaults
```

Liite 21. Raspberrypi3-konfiguraatiot SDN-testbed

```
root@raspberrypi-1:/home/pi# ovs-vsctl add-br of-switch
root@raspberrypi-1:/home/pi# ovs-vsctl add-port of-switch eth0
root@raspberrypi-1:/home/pi# ovs-vsctl add-port of-switch eth1
root@raspberrypi-1:/home/pi# ovs-vsctl add-port of-switch eth2
root@raspberrypi-1:/home/pi# ifconfig of-switch 172.16.1.103/24
root@raspberrypi-1:/home/pi# update-rc.d openvswitch-switch defaults
```